POLITECNICO DI TORINO

Corso di Laurea Magistrale in

Ingegneria Meccanica

Tesi di Laurea Magistrale

Progettazione e sviluppo di un prototipo di Ballbot

Relatore:

Prof. Stefano Pastorelli

Candidato:

Valerio Cornagliotto

Correlatore:

Prof. Stefano Mauro

A.A 2018/2019

Sommario

1 INTRODUZIONE

Nel seguente lavoro di tesi è stato effettuato un progetto di ballbot, un particolare robot in grado di muoversi e mantenersi in equilibrio sopra una sfera mediante l'attuazione di tre ruote omnidirezionali. Al fine di comprenderne il funzionamento, le caratteristiche strutturali nonché il comportamento cinematico e dinamico, è stato analizzato lo stato dell'arte mediante lo studio di alcune delle pubblicazioni riguardanti il robot. Inoltre, si è creato un modello prima analitico, implementandolo in un secondo momento in ambiente Matlab Simulink®. con lo scopo di simularne il comportamento reale è stata creata una rappresentazione grafica della dinamica tramite la piattaforma Mechanics Explorers. Questo aspetto si è dimostrato indispensabile in fase progettuale per effettuare alcune simulazioni prima di lavorare con il prototipo sperimentale. Dopo aver definito le componenti necessarie alla realizzazione del ballbot ed aver dimensionato opportunamente tutti i componenti, è stato creato un modello CAD mediante l'utilizzo del software SOLIDWORKS®. Infine è stato assemblato il prototipo e sono state condotte alcune prove cinematiche e di stabilità.

2 STATO DELL'ARTE

Esistono numerosi esempi di robot staticamente stabili, fondati sull'utilizzo di basi con 3 o 4 ruote, in grado di muoversi nello spazio mediante l'attuazione delle ruote. Un problema, spesso trascurato di questi robot, è la loro tendenza a divenire dinamicamente instabili con il rischio di ribaltamento. Per comprendere meglio la precaria stabilità di questi modelli, si analizza la dinamica di un esempio molto semplice di robot poggiante su 4 ruote.

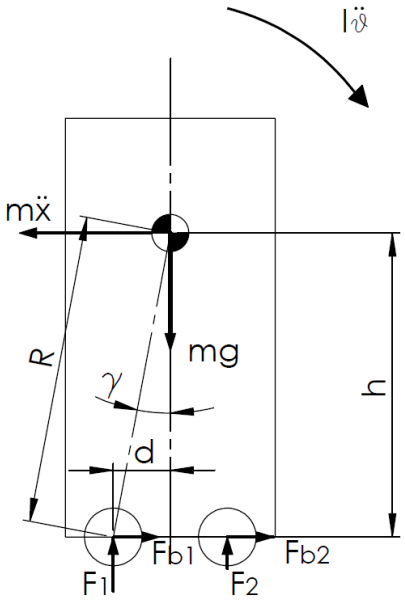


Figura 1: Diagramma di corpo libero robot staticaly stable

Scrivendo l'equilibrio di momento attorno al centro della ruota di sinistra, si ricava:

$$mh\ddot{x} - mg\big(d - R\vartheta cos(\gamma)\big) + 2F_2 d = I\ddot{\vartheta}$$

dove l'angolo γ rappresenta l'angolo tra la congiungente del centro ruota e il centro di massa con la verticale.

Durante la fase di frenata la forza di contatto F2 diminuisce fino ad essere nulla per un valore limite di decelerazione. L'equilibrio di momento si riduce alla formula seguente

$$\frac{mh\ddot{x} - mg\big(d - R\vartheta cos(\gamma)\big)}{I} = \ddot{\vartheta}$$

Se la decelerazione è tale che sia verificata la seguente disuguaglianza

$$h\ddot{x} > g\big(d - R\vartheta cos(\gamma)\big) \quad \longrightarrow \quad \ddot{\vartheta} > 0$$

si genera un'accelerazione angolare di ribaltamento $\ddot{\vartheta}$.

Considerando inoltre che all'aumentare dell'angolo ϑ diminuisce il braccio resistente al ribaltamento, una volta innescata la rotazione, se non diminuisce la decelerazione, il fenomeno è instabile con conseguente ribaltamento del robot. Analoga considerazione è valida in condizioni statiche qualora ci fosse una forza esterna applicata nel centro di massa invece della forza di inerzia.

$$F > \frac{mg\big(d - R\vartheta cos(\gamma)\big)}{h} \quad \longrightarrow \ddot{\vartheta} > 0$$

Per diminuire il fenomeno di ribaltamento è possibile intervenire abbassando il baricentro del robot, o aumentandone la larghezza. Tuttavia, per potersi muovere agevolmente in ambienti domestici e poter interagire con l'essere umano è necessario che siano snelli e alti abbastanza. Due elementi che influiscono negativamente sulla stabilità.

Un ulteriore punto a sfavore dell'utilizzo di modelli basati su 4 ruote, due delle quali sterzanti, è il raggio di sterzo e una limitata direzione di movimento con la necessità di doversi prima orientare secondo la direzione desiderata.

In Giappone, nella prima metà degli anni '90, sono stati sviluppati nuovi robot dinamicamente stabili mantenuti in equilibrio mediante l'attuazione di 2 ruote coassiali controllate da sistemi di controllo analoghi a quelli utilizzati per il pendolo inverso. Questa tecnologia è stata la base per lo sviluppo di mezzi di trasporto quali i Segway o i più recenti overboard (Figura 2).

Figura 2: Overbobard a sinistra; Segway a destra

Questi lavori hanno ispirato il gruppo di ricerca dell'istituto di robotica della Carnegie Mellon University capitanato dal Prof. Ralph Hollis, per lo sviluppo di un nuovo robot dinamicamente stabile in grado di rimanere in equilibrio e muoversi nello spazio su una sfera mediante l'attuazione di ruote opportunamente orientate.

2.1 Primo ballbot

Il primo ballbot proposto dalla Carnegie Mellon University, ha un'altezza di 1,41m e un peso di 55 kg. La sfera è una sfera cava in alluminio del diametro di 0.185m ricoperta da uno strato di 13mm di uretano per migliorare l'attrito tra sfera e terreno.

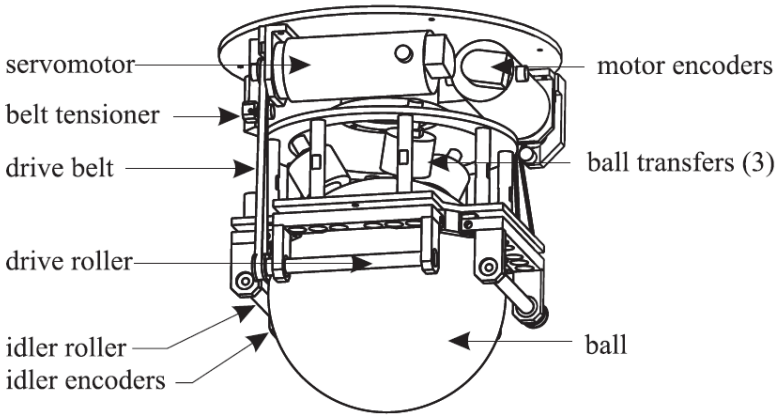


Figura 3: meccanismo di attuazione ballbot CMU

Il meccanismo di attuazione è simile a quello adoperato per leggere lo spostamento dei mouse a sfera. Due motori DC ad alta coppia sono collegati ad altrettanti rulli tramite una trasmissione a cinghia. I rulli sono posti in prossimità della sezione maggiore di sfera e orientati ortogonalmente tra loro con entrambi gli assi appartenenti ad un piano parallelo al terreno.

Questa configurazione dei rulli di attuazione permette la rotazione della sfera attorno a qualsiasi asse appartenente al piano contenente gli assi dei rulli attuatori.

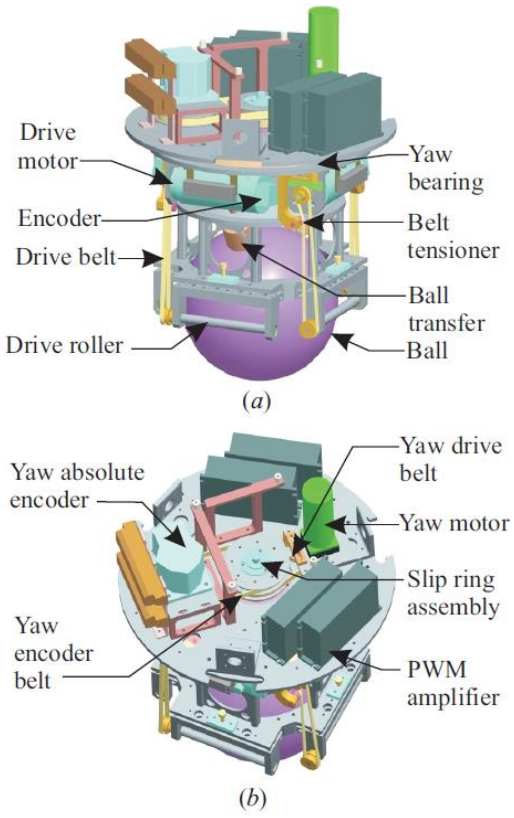


Figura 4: meccanismo di attuazione della sfera (a), meccanismo di attuazione del corpo robot (b).

La rotazione (yaw) del corpo del robot attorno all'asse Z perpendicolare al terreno, è ottenuto aggiungendo un ulteriore grado di libertà tra la zona di attuazione, rappresentata in Figura 3, e il corpo robot. Il giunto è formato da un cuscinetto sottile assiale e un motore DC collegato a un riduttore epicicloidale (Figura 4 (b)). Collegando la sezione superiore del robot al portatreno del riduttore e la parte inferiore alla corona, attuando il pignone centrale del riduttore si ottiene la rotazione del corpo robot con un elevato rapporto di riduzione.

La prima versione del robot proposto presentava 2 rulli attivi di attuazione e due rulli passivi diametralmente opposti ai primi collegati agli encoder. Questo produceva una coppia resistente sulla sfera che causava un movimento "saltellante" indesiderato. Per questo motivo i due rulli passivi sono stati resi attivi con l'aggiunta di due ulteriori motori DC.

2.2 Modelli successivi

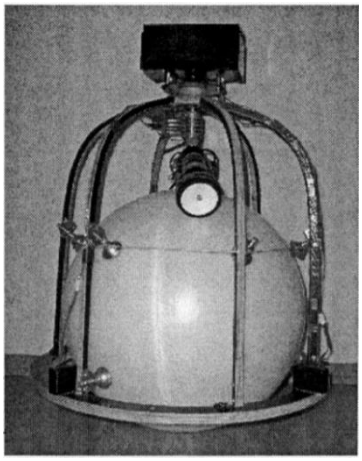


Figura 5: ERROSphere

Successivamente alla presentazione del ballbot da parte della CMU, altri gruppi di ricerca si sono cimentati nello studio e nella realizzazione di robot auto-bilanciati in equilibrio su una sfera. Nel 2005 l'ungherese Laszlo Havasi sviluppò un ballbot chiamato ERROSphere, rappresentato in Figura 5, il cui bilanciamento tuttavia non risultò particolarmente affidabile e il prototipo non ebbe ulteriori sviluppi.

Nel 2008 Masaaki Kumagai sviluppò un ballbot chiamato BallIP (Figura 7) alla Tohoku Gakuin University. L'anno successivo alla presentazione del robot, il gruppo di ricerca dimostrò come il robot fosse in grado di collaborare con l'essere umano nel trasporto di carichi, anche mediante la collaborazione di più robot (Figura 6).

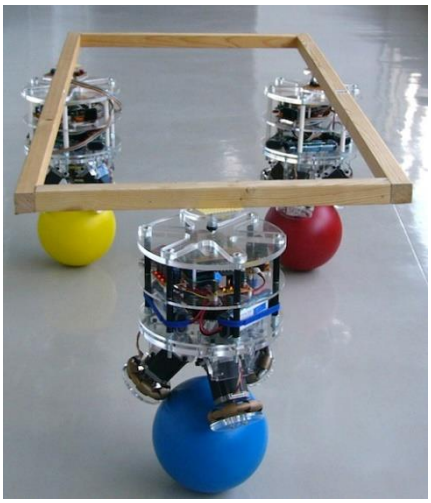


Figura 6: Collaborazione di tre BallIP per la movimentazione di un carico

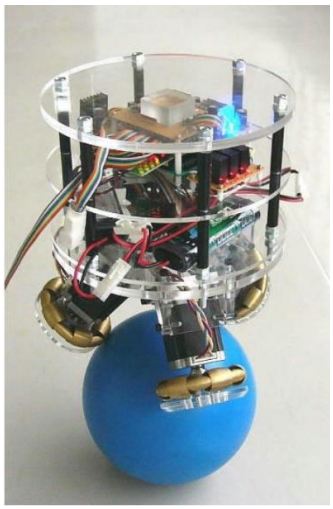


Figura 7: The BallIP

Nel 2010 un gruppo di studenti di ingegneria meccanica del università ETH di Zurigo, sviluppò un ballbot chiamato Rezero.

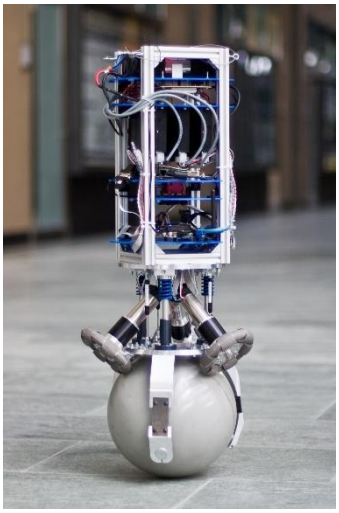


Figura 8: Rezero

Questo prototipo vide un successivo sviluppo che portò a un vero e proprio robot di design con elevate prestazioni dinamiche grazie ad un accurato sistema di controllo e all'utilizzo di motori brushless ad elevate prestazioni.

La differenza dei successivi prototipi di ballbot funzionanti, rispetto al modello proposto dalla CMU, è il sistema di attuazione della sfera. Mentre il sistema, già analizzato in precedenza, del ballbot del Prof. Hollis, si basa su un sistema di tipo mouse inverso, i modelli di BallIP e Rezero hanno un sistema di attuazione mediante tre ruote omnidirezionali posizionate a 120° l'una dall'altra. Questo sistema permette di ottenere la rotazione di "yaw", ovvero la rotazione attorno all'asse Z, semplicemente attuando tutte e tre le ruote contemporaneamente, a differenza di quanto avviene nel ballbot della CMU nel quale è presente un ulteriore motore di attuazione per la rotazione del robot. Inoltre, l'utilizzo di ruote omnidirezionali, elimina il fenomeno di strisciamento che avviene tra i rulli e la sfera.

3 MODELLO DINAMICO

In questa sezione è descritto il modello dinamico del robot studiato come un pendolo inverso in due piani perpendicolari indipendenti tra loro.

Questo risulta un modello semplificato rispetto il modello 3D poiché esistono delle forze dipendenti dallo stato di moto nel piano perpendicolare. Risulta comunque un'approssimazione accettabile ai fini del calcolo di alcuni dati significativi per le scelte dei componenti, ad esempio per la stima delle coppie richieste ai motori e del coefficiente di attrito tra ruota e sfera atto a garantire un moto di rotolamento puro tra le ruote e la sfera.

Il modello creato è utilizzato anche per definire un semplice controllo PD del ballbot. Infatti, leggendo le grandezze di feedback nei due piani distinti, è possibile calcolare l'accelerazione da fornire alla sfera per mantenere l'equilibrio verticale e generare gli spostamenti sul piano della stessa.

Tuttavia è necessario considerare che lruote e i motori, ad esempio, appartengono a tre piani distinti inclinati di 120° l'uno dall'altro, pertanto, i vettori di velocità angolare e i momenti di inerzia degli stessi, appartengono a loro volta a tre piani distinti. È necessario dunque creare un modello che riporti tali grandezze su due piani perpendicolari, a tal fine è stato creato un modello equivalente che preveda la presenza di una singola ruota motrice per ognuno dei due piani considerati.

3.1 Modello equivalente

Al fine di semplificare il modello si crea un modello avente una sola ruota equivalente ai motori e le tre ruote (W), posta sopra la sfera, e una massa sospesa (P), equivalente del telaio del robot.

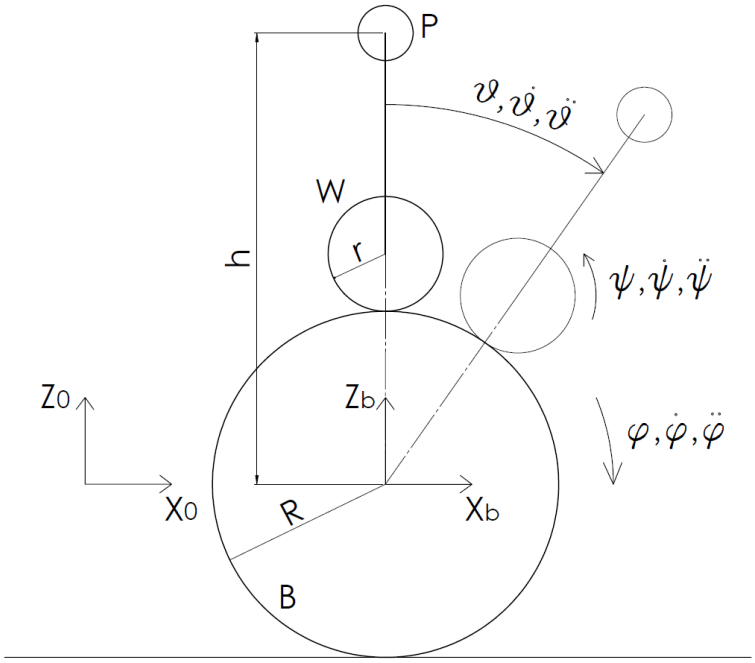


Figura 9: Modello equivalente del ballbot

Si definiscono le coordinate del centro sfera B, del centro della ruota equivalente W e della massa sospesa P.

$$x_b = \varphi \cdot R$$
$$z_b = 0$$

$$x_w = x_b + (R + r) \cdot sin(\vartheta)$$
$$z_w = (R + r) \cdot cos(\vartheta)$$

$$x_p = x_b + h \cdot sin(\vartheta)$$
$$z_p = h \cdot cos(\vartheta)$$

La velocità di rotazione della ruota equivalente[1] è ricavata imponento la condizione di rotolamento puro della ruota sulla sfera. Per valutare questa condizione si ricava la velocità assoluta del punto di contatto tra i due corpi, considerandolo prima fisso alla sfera (punto D in Figura 10) e poi fisso alla ruota (punto C in Figura 10)

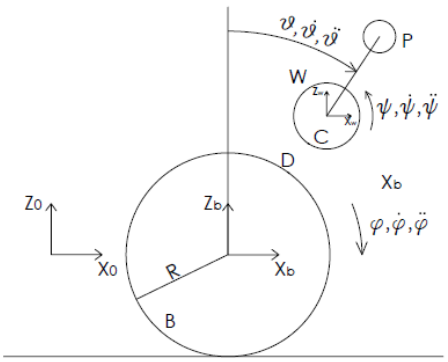


Figura 11: Condizione di rotolamento puro della ruota equivalente

Ponendo l'uguaglianza tra le velocità così ottenute è possibile valutare $\dot{\psi}$ in funzione di $\dot{\varphi}$ e $\dot{\vartheta}$.

$$\vec{v}_D = \vec{v}_{tras.} + \vec{v}_{rel.} =$$
$$= \begin{bmatrix} \dot{x}_B \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\varphi} \\ 0 \end{bmatrix} \times \begin{bmatrix} Rsin(\vartheta) \\ 0 \\ Rcos(\vartheta) \end{bmatrix}$$

[1] Si noti in Figura 10 che è stata scelta una convenzione di rotazione della ruota tale per cui una rotazione positiva della ruota produca una rotazione positiva della sfera.

Sviluppando il prodotto vettoriale:

$$\begin{bmatrix}0\\ \dot{\varphi}\\ 0\end{bmatrix}\times\begin{bmatrix}Rsin(\vartheta)\\ 0\\ Rcos(\vartheta)\end{bmatrix}=det\begin{bmatrix}i & j & k\\ 0 & \dot{\varphi} & 0\\ Rsin(\vartheta) & 0 & Rcos(\vartheta)\end{bmatrix}$$

Risulta pertanto:

$$\vec{v}_D=\begin{bmatrix}\dot{x}_B+\dot{\varphi}\cdot Rcos(\vartheta)\\ 0\\ -\dot{\varphi}\cdot Rcos(\vartheta)\end{bmatrix}$$

Analogamente si scrive la velocità assoluta del punto C:

$$\vec{v}_C=\vec{v}_{tras.}+\vec{v}_{rel.}=$$
$$=\begin{bmatrix}\dot{x}_B\\ 0\\ 0\end{bmatrix}+\begin{bmatrix}0\\ \dot{\vartheta}\\ 0\end{bmatrix}\times\begin{bmatrix}(R+r)\cdot sin(\vartheta)\\ 0\\ (R+r)\cdot cos(\vartheta)\end{bmatrix}+\begin{bmatrix}0\\ -\dot{\psi}\\ 0\end{bmatrix}\times\begin{bmatrix}-r\cdot sin(\vartheta)\\ 0\\ -r\cdot cos(\vartheta)\end{bmatrix}$$

La velocità di trascinamento in questo caso è la composizione della velocità del sistema di riferimento B con la velocità del sistema di riferimento W, il quale mantiene l'orientazione con Xw orizzontale e Zw verticale e ruota attorno al sistema di riferimento B.

Risolvendo i prodotti vettoriali si ottiene:

$$\vec{v}_C=\begin{bmatrix}\dot{x}_B+\dot{\vartheta}\cdot(R+r)cos(\vartheta)+\dot{\psi}\cdot rcos(\vartheta)\\ 0\\ -\dot{\vartheta}\cdot(R+r)sin(\vartheta)-\dot{\psi}\cdot r\cdot sin(\vartheta)\end{bmatrix}$$

È ora possibile imporre la condizione di aderenza uguagliando le velocità.

$$\begin{cases}\dot{x}_B+\dot{\varphi}\cdot Rcos(\vartheta)=\dot{x}_B+\dot{\vartheta}\cdot(R+r)cos(\vartheta)+\dot{\psi}\cdot rcos(\vartheta)\\ \dot{\varphi}\cdot Rcos(\vartheta)=\dot{\vartheta}\cdot(R+r)sin(\vartheta)+\dot{\psi}\cdot r\cdot sin(\vartheta)\end{cases}$$

Risulta infine:

$$\dot{\psi}=\frac{R}{r}(\dot{\varphi}-\dot{\vartheta})-\dot{\vartheta}$$

È possibile verificare l'equazione immaginando che la ruota sia ferma e il punto di contatto ruota sfera non cambi, ruota e sfera si comportano un unico corpo rigido. Imponendo una

rotazione della sfera risulterebbe verificata l'uguaglianza $\dot{\varphi} = \dot{\vartheta}$ e l'equazione precedente si semplificherebbe in:

$$\dot{\psi} = -\dot{\vartheta}$$

Integrando nel tempo entrambi i membri si ricava $\psi = -\vartheta$.

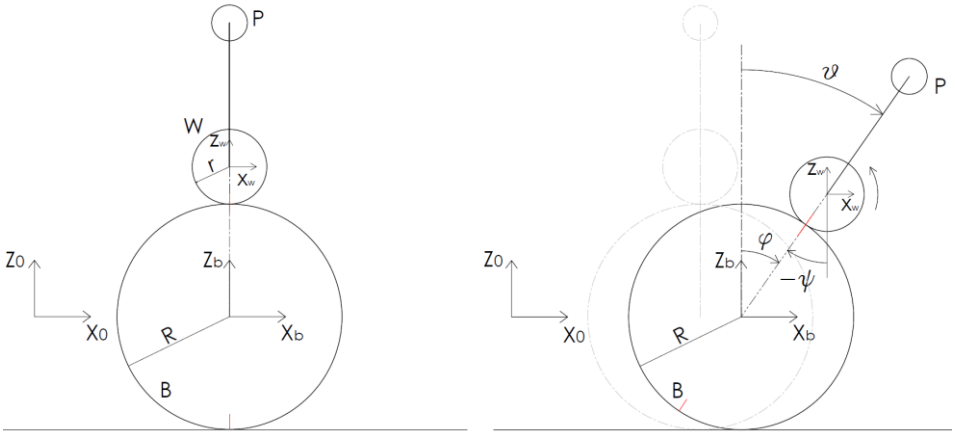


Figura 12: Relazione tra ψ, ϑ e ϕ.

3.2 Equazioni del moto

Al fine di determinare le equazioni del moto si è scelto di utilizzare un approccio di tipo lagrangiano per la risoluzione di sistemi dinamici, considerando il modello equivalente descritto in precedenza e analizzando distintamente i due piani XZ e YZ. L'equazione da risolvere è l'equazione di Eulero-Lagrange

$$\frac{\partial}{\partial t}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \vec{f}_{NP}$$

Dove L *,* Ovvero la differenza tra energia cinetica e energia potenziale del sistema, rappresenta la Lagrangiana, q_i sono le coordinate generalizzate di Lagrange ($\varphi\ e\ \vartheta$)[2] e f_{NP} è la risultante delle forze non potenziali.

$$\mathcal{L} = T - U$$

Per ricavare l'energia cinetica e potenziale del sistema si calcolano le singole energie dei corpi sfera, ruota e massa sospesa e si applica il principio di sovrapposizione degli effetti.

$$\mathrm{T} = T_b + T_w + T_p$$
$$U = U_b + U_w + U_p$$

Si descrive il procedimento utilizzato in entrambi i piani. Si procede quindi ricavando le energie cinetiche e potenziali per ogni corpo presente nel sistema; in seguito, dopo aver sostituito la lagrangiana meccanica, si risolve l'equazione di Eulero-Lagrange ricavando le equazioni del moto.

[2] È necessario riferire tutte le coordinate in funzione delle coordinate generalizzate scelte per descrivere il sistema. Nel nostro caso abbiamo già ricavato in precedenza l'equazione di ψ in funzione di ϑ e ϕ, pertanto è sufficiente sostituirla.

Sfera:

$$T_b = \frac{1}{2} m_b (R\dot{\varphi})^2 + \frac{1}{2} I_b \dot{\varphi}^2$$

$$U_b = 0$$

Ruota:

$$T_w = \frac{1}{2} m_w \left(\dot{x}_w \cdot \dot{x}_w{}^T \right) + \frac{1}{2} I_w \dot{\psi}^2$$

Essendo:

$$x_w = \begin{bmatrix} R\varphi + (R+r)\sin\vartheta \\ 0 \\ (R+r)\cos\vartheta \end{bmatrix} \overset{\frac{d}{dt}}{\rightarrow} \dot{x}_w = \begin{bmatrix} R\dot{\varphi} + (R+r)\cos\vartheta\,\dot{\vartheta} \\ 0 \\ -(R+r)\sin\vartheta\,\dot{\vartheta} \end{bmatrix}$$

Sostituendo nell'equazione risulta:

$$T_w = \frac{1}{2} m_w \begin{bmatrix} R\dot{\varphi} + (R+r)\cos\vartheta\,\dot{\vartheta} \\ 0 \\ -(R+r)\sin\vartheta\,\dot{\vartheta} \end{bmatrix}^2 + \frac{1}{2} I_w \left(\frac{R}{r}\left(\dot{\varphi} - \dot{\vartheta}\right) - \dot{\vartheta} \right)^2 =$$

$$= \frac{1}{2} m_w \left((R\dot{\varphi})^2 + 2R\dot{\varphi}(R+r)\cos\vartheta\,\dot{\vartheta} + (R+r)^2\cos^2(\vartheta)\,\dot{\vartheta}^2 + (R+r)^2\sin^2(\vartheta)\,\dot{\vartheta}^2 \right)$$

$$+ \frac{1}{2} I_w \left(\frac{R}{r}\left(\dot{\varphi} - \dot{\vartheta}\right) - \dot{\vartheta} \right)^2 =$$

$$= \frac{1}{2} m_w \left((R\dot{\varphi})^2 + 2R\dot{\varphi}(R+r)\cos\vartheta\,\dot{\vartheta} + (R+r)^2\dot{\vartheta}^2 \right) + \frac{1}{2} I_w \left(\frac{R}{r}\left(\dot{\varphi} - \dot{\vartheta}\right) - \dot{\vartheta} \right)^2$$

$$U_w = m_w g (R+r)\cos\vartheta$$

Massa sospesa:

$$T_p = \frac{1}{2} m_p \left(\dot{x}_p \cdot \dot{x}_p{}^T \right) + \frac{1}{2} I_p \dot{\vartheta}^2$$

Essendo:

$$x_p = \begin{bmatrix} R\varphi + h \sin\vartheta \\ 0 \\ h \cos\vartheta \end{bmatrix} \overset{\frac{d}{dt}}{\rightarrow} \dot{x}_p = \begin{bmatrix} R\dot{\varphi} + h \cos\vartheta \, \dot{\vartheta} \\ 0 \\ -h \sin\vartheta \, \dot{\vartheta} \end{bmatrix}$$

Sostituendo nell'equazione risulta:

$$T_p = \frac{1}{2} m_p \begin{bmatrix} R\dot{\varphi} + h \cos\vartheta \, \dot{\vartheta} \\ 0 \\ -h \sin\vartheta \, \dot{\vartheta} \end{bmatrix}^2 + \frac{1}{2} I_p \dot{\vartheta}^2 =$$

$$= \frac{1}{2} m_w \left((R\dot{\varphi})^2 + 2R\dot{\varphi} h \cos\vartheta \, \dot{\vartheta} + h^2 \cos^2(\vartheta) \, \dot{\vartheta}^2 + h^2 \sin^2(\vartheta) \, \dot{\vartheta}^2 \right) + \frac{1}{2} I_p \dot{\vartheta}^2 =$$

$$= \frac{1}{2} m_p \left((R\dot{\varphi})^2 + 2R\dot{\varphi} h \cos\vartheta \, \dot{\vartheta} + h^2 \dot{\vartheta}^2 \right) + \frac{1}{2} I_p \dot{\vartheta}^2$$

$$U_p = m_p g h \cos\vartheta$$

Essendo $q_i = \begin{bmatrix} \varphi \\ \vartheta \end{bmatrix}$ si possono risolvere le equazioni di Lagrange.

$$\frac{\partial}{\partial t} \left(\frac{\partial T}{\partial \dot{q}_i} \right) - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} = \vec{f}_{NP}$$

Derivazione per φ:

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\mathrm{m_b R^2\dot{\varphi} + I_b\dot{\varphi} + m_w R^2\dot{\varphi} + m_w R(R+r)\cos\vartheta\dot{\vartheta} + \frac{1}{2}I_w\left(2\left(\frac{R}{r}\right)^2(\dot{\varphi}-\dot{\vartheta}) - 2\left(\frac{R}{r}\right)\dot{\vartheta}\right)}\right.$$
$$\left. + \mathrm{m_p R^2\dot{\varphi} + m_p Rh\cos\vartheta\,\dot{\vartheta}}\right) = \vec{f}_{\mathrm{NP}_{(1,1)}}$$

$$\mathrm{m_b R^2\ddot{\varphi} + I_b\ddot{\varphi} + m_w R^2\ddot{\varphi} + m_w R(R+r)\cos\vartheta\ddot{\vartheta} - m_w R(R+r)\sin\vartheta\,\dot{\vartheta}^2}$$
$$+ \mathrm{I_w}\left(\left(\frac{R}{r}\right)^2(\ddot{\varphi}-\ddot{\vartheta}) - \left(\frac{R}{r}\right)\ddot{\vartheta}\right) + \mathrm{m_p R^2\ddot{\varphi} + m_p Rh\cos\vartheta\,\ddot{\vartheta} - m_p Rh\sin\vartheta\,\dot{\vartheta}^2} = \vec{f}_{\mathrm{NP}_{(1,1)}}$$

$$\boxed{\begin{gathered}\ddot{\varphi}\left(\left(m_b + m_w + m_p\right)R^2 + \mathrm{I_b} + \mathrm{I_w}\left(\frac{R}{r}\right)^2\right) + \\ +\ddot{\vartheta}\left(\mathrm{m_w R(R+r)}\cos\vartheta - \left(\frac{R}{r}\right)^2 - \left(\frac{R}{r}\right) + \mathrm{m_p Rh}\cos\vartheta\right) + \\ -\left(\mathrm{m_w(R+r) + m_p h}\right)R\sin\vartheta\,\dot{\vartheta}^2 = \vec{f}_{\mathrm{NP}_{(1,1)}}\end{gathered}}$$

Derivazione per ϑ:

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\mathrm{m_w R\dot{\varphi}(R+r)\cos\theta + m_w(R+r)^2\dot{\vartheta} + \frac{1}{2}I_w\left(4\left(\frac{R}{r}\right)^2\dot{\vartheta} - 2\left(\frac{R}{r}\right)^2\dot{\varphi} - 2\left(\frac{R}{r}\right)\dot{\varphi} - 4\left(\frac{R}{r}\right)\dot{\vartheta} + 2\dot{\vartheta}\right)}\right.$$
$$\left. + \mathrm{m_p\dot{\varphi}h\cos\theta + m_p h^2\dot{\vartheta}\ + I_p\dot{\vartheta}}\right) + \mathrm{m_w R\dot{\varphi}(R+r)\dot{\vartheta}\sin\vartheta + m_p R\dot{\varphi}h\dot{\vartheta}\sin\vartheta} = \vec{f}_{\mathrm{NP}_{(2,1)}}$$

$$\mathrm{m_w R\ddot{\varphi}(R+r)\cos\vartheta} - \cancel{\mathrm{m_w R\dot{\varphi}(R+r)\dot{\vartheta}\sin\vartheta}} + \mathrm{m_w(R+r)^2\ddot{\vartheta}}$$
$$+ \mathrm{I_w}\left(\left(\frac{R}{r}\right)^2(\ddot{\vartheta}-\ddot{\varphi}) - \left(\frac{R}{r}\right)(\ddot{\varphi}-2\ddot{\vartheta}) + \ddot{\vartheta}\right) + \mathrm{m_p R\ddot{\varphi}h\cos\vartheta} - \cancel{\mathrm{m_p R\dot{\varphi}h\,\dot{\vartheta}\sin\vartheta}} + \mathrm{m_p h^2\ddot{\vartheta} + I_p\ddot{\vartheta}}$$
$$\cancel{+\mathrm{m_w R\dot{\varphi}(R+r)\dot{\vartheta}\sin\vartheta}} + \cancel{\mathrm{m_p R\dot{\varphi}h\dot{\vartheta}\sin\vartheta}} - \mathrm{(m_w(R+r) + m_p h)g\sin\vartheta} = \vec{f}_{\mathrm{NP}_{(2,1)}}$$

$$\boxed{\begin{gathered}\ddot{\varphi}\left(m_w R(R+r)\cos\vartheta - I_w\left(\frac{R}{r}\right)^2 - I_w\left(\frac{R}{r}\right) + m_p Rh\cos\vartheta\right) + \\ +\ddot{\vartheta}\left(\left(\frac{R}{r}\right)^2 I_w + I_w + \frac{2R}{r}I_w + m_w(R+r)^2 + m_p h^2 + I_p\right) + \\ -\left(\mathrm{m_w(R+r) + m_p h}\right)\mathrm{g}\sin\vartheta = \vec{f}_{\mathrm{NP}_{(2,1)}}\end{gathered}}$$

Nelle equazioni scritte si è lasciato scritto in forma implicita le forze non potenziale. Pertanto è necessario a questo punto definire tali forze generalizzate e sostituirle nelle equazioni.

La coppia non potenziale nel ballbot, è la coppia generata dai motori e trasferita alla sfera; tuttavia, nel sistema equivalente, la coppia è generata da una singola ruota, pertanto, si considera coppia generalizzata unicamente il contributo della singola ruota; in seguito sarà descritto il procedimento per riferire tale coppia agli assi motori.

La coppia fornita dalla ruota equivalente, agisce direttamente sul grado di libertà ψ, il quale tuttavia, non è una coordinata generalizzata scelta per descrivere il sistema, pertanto è necessario tradurre l'effetto che la coppia produce sulle due coordinate scelte φ e ϑ.

Avendo già ricavato:

$$\begin{bmatrix} 0 \\ -\dot{\psi} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{R}{r}\left(\dot{\varphi} - \dot{\vartheta}\right) + \dot{\vartheta} \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 \\ -\frac{R}{r} & \left(\frac{R}{r}+1\right) \\ 0 & 0 \end{bmatrix}}_{J} \times \begin{bmatrix} \dot{\varphi} \\ \dot{\vartheta} \end{bmatrix}$$

Analogamente:

$$\vec{M} = \begin{bmatrix} 0 \\ -M \\ 0 \end{bmatrix}$$

$$\vec{\mathrm{f}}_{\mathrm{NP},1} = J^T \times \vec{M}$$

Potendo semplificare la matrice jacobiana eliminando i gradi di libertà corrispondenti alle righe nulle risulta:

$$\vec{\mathrm{f}}_{\mathrm{NP},1} = \underbrace{\begin{bmatrix} -\frac{R}{r} \\ \left(\frac{R}{r}+1\right) \end{bmatrix}}_{J^T} \times (-M) = \begin{bmatrix} \frac{R}{r}M \\ -\left(\frac{R}{r}+1\right)M \end{bmatrix}$$

La coppia di reazione che si genera agisce direttamente sulla coordinata ϑ.

$$\begin{bmatrix} 0 \\ \vartheta \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}_{J} \times \begin{bmatrix} \dot{\varphi} \\ \dot{\vartheta} \end{bmatrix}$$

$$\vec{f}_{NP,2} = \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{J^T} \times (M) = \begin{bmatrix} 0 \\ M \end{bmatrix}$$

La risultante delle forze non potenziali risulta:

$$\vec{f}_{NP} = \begin{bmatrix} \vec{f}_{NP_{(1,1)}} \\ \vec{f}_{NP_{(2,1)}} \end{bmatrix} = \begin{bmatrix} \frac{R}{r} M \\ -\left(\frac{R}{r} + 1\right) M + M \end{bmatrix} = \begin{bmatrix} \frac{R}{r} M \\ -\frac{R}{r} M \end{bmatrix}$$

È possibile riscrivere il sistema non lineare ottenuto, in forma matriciale come segue:

$$M(\vec{q}, \dot{\vec{q}})\ddot{q} + C(\vec{q}, \dot{\vec{q}}) + G(\vec{q}) = \vec{f}_{NP}$$

Definendo:

$$\gamma = m_w(R + r) + m_p h$$
$$m_{tot} = m_p + m_b + m_w$$

Risulta:

$$M = \begin{bmatrix} m_{tot}R^2 + I_b + I_w\left(\frac{R}{r}\right)^2 & \gamma R\cos\vartheta + I_w\frac{R}{r^2}(R + r) \\ \gamma R\cos\vartheta - I_w\frac{R}{r^2}(R + r) & I_w\frac{(R + r)^2}{r^2} + m_w(R + r)^2 + I_p + m_p h^2 \end{bmatrix}$$

$$C = \begin{bmatrix} -\gamma R\sin\vartheta\,\dot{\vartheta}^2 \\ 0 \end{bmatrix}$$

$$G = \begin{bmatrix} 0 \\ -\gamma g\sin\vartheta \end{bmatrix}$$

3.3 Calcolo inerzie

Nelle equazioni del moto sono presenti più termini che si riferiscono ai momenti di inerzia dei corpi in esame.

I_b: momento di inerzia della sfera

I_w: momento di inerzia della ruota equivalente

I_p: momento di inerzia delle masse sospese

Per quanto riguarda la sfera, il momento di inerzia è calcolato considerando un guscio sferico, in quanto lo spessore è trascurabile rispetto il raggio.

Risulta pertanto:

$$I_b = \frac{2}{3} m_b R^2$$

Il momento di inerzia relativo alle masse sospese è stato in prima analisi approssimato a quello di una massa concentrata nel baricentro delle masse sospese e calcolato come segue:

$$I_p = m_p h^2$$

In seconda analisi è stato sostituito tale valore con il valore estratto dalla matrice di inerzia del modello CAD, ricavata mediante Solidworks®.

$$I = \begin{bmatrix} 0.36 & 0 & 0 \\ 0 & 0.36 & 0 \\ 0 & 0 & 0.06 \end{bmatrix}$$

Tale matrice è calcolata rispetto il sistema di coordinate centrato nella sfera e avente asse x con verso in direzione del motore 1 (Definito R_0 nel paragrafo "*5.2 Blocco cinematica vs.* dinamica" del capitolo "5 Modello simulink 3D").

L'inerzia relativa alla ruota equivalente introdotta nel modello viene calcolata uguagliando l'inerzia relativa al sistema ruota-motore, del sistema reale all'inerzia del sistema equivalente.

L'inerzia delle omniwheel è calcolata considerando la ruota come un cilindro. Risulta:

$$I_{ow} = \frac{1}{2} m_w r^2$$

Essendo I_{ow} l'inerzia relativa alla ruota reale.

Occorre proiettare le velocità di rotazione delle tre ruote nei piani XZ e YZ per poterle eguagliare al sistema equivalente introdotto dal modello.

Considerando lo spostamento in direzione X:

$$\omega_1 = 0$$

$$\omega_2 = \frac{\sqrt{3}}{2} \dot{\psi}_x \cdot \cos\alpha$$

$$\omega_3 = -\frac{\sqrt{3}}{2} \dot{\psi}_x \cdot \cos\alpha$$

In direzione Y:

$$\omega_1 = \dot{\psi}_y \cdot \cos\alpha$$

$$\omega_{2/3} = \frac{1}{2} \dot{\psi}_y \cdot \cos\alpha$$

Ora è possibile scrivere l'equilibrio dei momenti di inerzia:

Direzione X:

$$\frac{1}{2} I_w \dot{\psi}_x{}^2 = 2 \cdot \left(\frac{1}{2} (I_{ow} + I_m) \left(\frac{\sqrt{3}}{2} \dot{\psi}_x \cdot \cos\alpha \right)^2 \right)$$

$$I_w = \frac{3}{2} \cos^2(\alpha) \, (I_{ow} + I_m)$$

Direzione Y:

$$\frac{1}{2} I_w \dot{\psi}_y{}^2 = \frac{1}{2} (I_{ow} + I_m) \left(\dot{\psi}_y \cdot \cos\alpha \right)^2 + 2 \cdot \left(\frac{1}{2} (I_{ow} + I_m) \left(\frac{1}{2} \dot{\psi}_y \cdot \cos\alpha \right)^2 \right)$$

$$I_w = \frac{3}{2} \cos^2(\alpha) \, (I_{ow} + I_m)$$

Essendo l'inerzia del motore fornita dal costruttore, e l'inerzia delle ruote, calcolata come descritto nella pagina precedente

$$I_m = 4.4\text{e} - 5\ \text{Kg} \cdot \text{m}^2$$

$$I_{ow} = 3.625\text{e} - 4\ \text{Kg} \cdot \text{m}^2$$

Risulta:

$$I_w = 3.26e - 4\ \text{Kg} \cdot \text{m}^2$$

4 MODELLO CINEMATICO

Le ruote omnidirezionali o omni-wheel sono delle ruote in grado di trasferire il moto senza che la direzione della velocità della base rispetto la ruota, nel punto di contatto base-ruota, corrisponda necessariamente alla velocità tangenziale della ruota misurata nel medesimo punto. Sostanzialmente sono in grado di muoversi in una direzione diversa dalla direzione imposta dal rotolamento della ruota, senza strisciamento relativo con la base ma sfruttando il rotolamento dei rullini presenti sulla ruota.

Esistono diverse tipologie di omni-wheel:

A singola fila di rullini: questa tipologia presenta una sola fila di rullini che permetto lo spostamento in direzione parallela all'asse della ruota. Questo tipo di ruota introduce delle vibrazioni dovute al cambiamento del diametro della ruota nello spazio presente tra 2 rullini consecutivi, ma il punto di contatto della ruota con la base resta invariato durante il moto. Il problema delle vibrazioni è stato affrontato riducendo il gap tra i rullini oppure disegnando i rullini in modo da rendere più graduale il cambio di diametro; altre soluzioni prevedono la presenza di una sfera nel gap. Un ulteriore modello di ruota omnidirezionale chiamata Mecanum wheel presenta i rullini inclinati di 45° rispetto l'asse. Questa soluzione riduce il problema delle vibrazioni causate dal gap tra i rulli, ma introduce problemi di slittamento e vibrazioni laterali.

A due file di rullini: questa versione di ruota omnidirezionale presenta due file di rulli sfasati in modo tale che nell'istante in cui il contatto sia in corrispondenza del rullino in una fila, l'altra fila presenti lo spazio tra i rullini. In questo modo il contatto è continuo e il diametro della ruota non cambia, a cambiare è invece il punto di contatto che si sposta da una fila all'altra con una frequenza pari a $\frac{2N*n}{60}$ dove N è il numero di rullini per ogni fila e n è la velocità espressa in rpm.

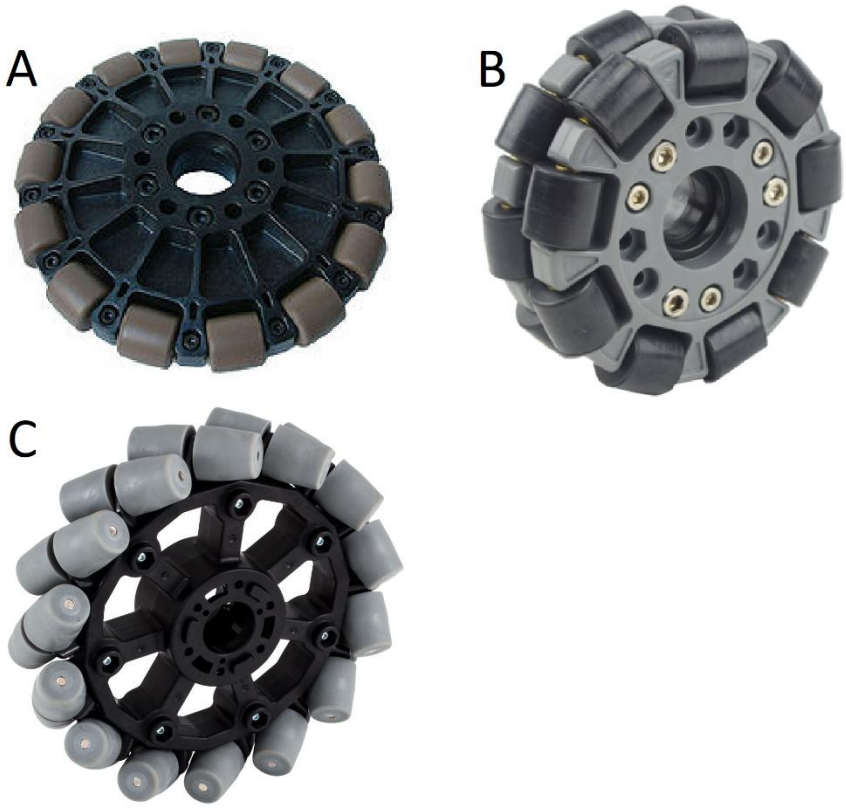


Figura 13: (A) Omniwheel a singola fila di rulli, (B) Omniwheel a due file di rulli, (C) Mecanum wheel

4.1 Modello cinematico approssimato

In questa sezione si sviluppa il calcolo della matrice jacobiana che lega la velocità angolare della sfera alla velocità delle ruote considerando il punto di contatto ruota sfera come il punto mediano della ruota, cosa che effettivamente avviene nel caso di ruota omnidirezionale a singola fila.

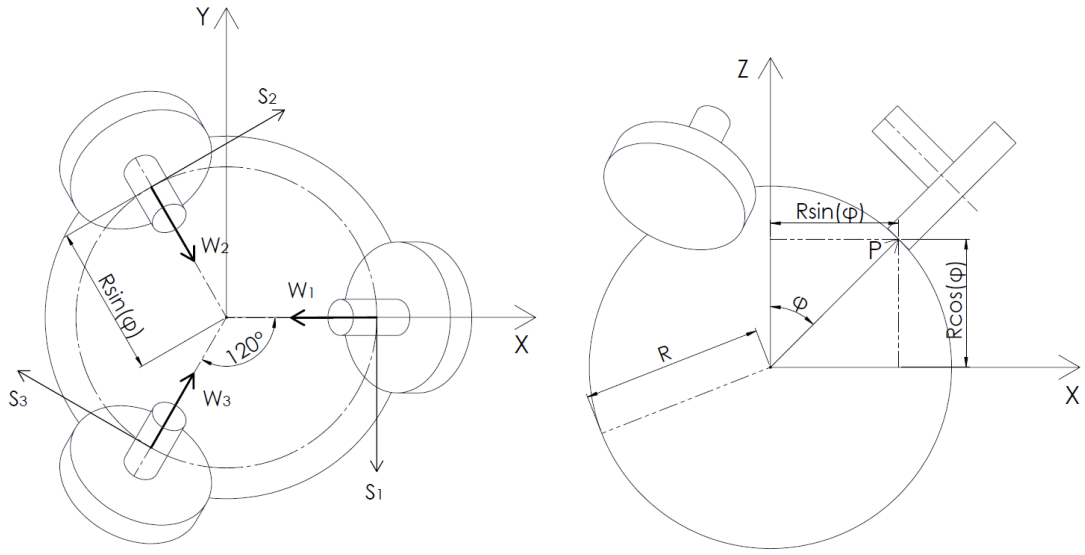


Figura 14: geometria delle ruote sulla sfera

Essendo la velocità v_{si} la velocità tangenziale della sfera nel punto di contatto della ruota i-esima, risulta:

$$\bar{v}_{si} = \bar{\omega} \times \bar{p}_i$$

dove ω rappresenta la velocità angolare della ruota e p_i il vettore posizione del punto di contatto della ruota i-esima a partire dal centro sfera. In riferimento alla Figura 14 si possono definire i vettori posizione del punto di contatto.

$$\bar{p}_1 = (\mathrm{Rsin}(\varphi), \quad 0, \quad \mathrm{Rcos}(\varphi))$$

$$\bar{p}_2 = \left(-\mathrm{Rsin}(\varphi)\sin(\gamma), \mathrm{Rsin}(\varphi)\cos(\gamma), \mathrm{Rcos}(\varphi)\right) =$$

$$= \left(-\frac{1}{2}\mathrm{Rsin}(\varphi), \quad \frac{\sqrt{3}}{2}\mathrm{Rsin}(\varphi), \quad \mathrm{Rcos}(\varphi)\right)$$

$$\bar{p}_3 = \left(-\mathrm{Rsin}(\varphi)\sin(\gamma), -\mathrm{Rsin}(\varphi)\cos(\gamma), \mathrm{Rcos}(\varphi)\right) =$$

$$= \left(-\frac{1}{2}\mathrm{Rsin}(\varphi), \quad \frac{\sqrt{3}}{2}\mathrm{Rsin}(\varphi), \quad \mathrm{Rcos}(\varphi)\right)$$

Per conoscere la velocità tangenziale delle ruote $\bar{v}_{wi}$ bisogna calcolare la proiezione della velocità tangenziale della sfera nel punto di contatto con la ruota e il versore della velocità tangenziale delle ruote s_i

$$\bar{v}_{wi} = \bar{v}_{si} \cdot \bar{s}_i$$

essendo inoltre la velocità tangenziali delle ruote pari $\overline{v_{wi}} = \overline{\omega_i} \cdot \bar{r}$, risulta:

$$\bar{\omega}_i = \frac{1}{r}(\bar{\omega} \times \bar{p}_i) \cdot \bar{s}_i$$

infine, risulta:

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \frac{1}{r}\cdot\begin{bmatrix} Rcos(\varphi) & 0 & -Rsin(\varphi)\\ -\frac{1}{2}Rcos(\varphi) & \frac{\sqrt{3}}{2}Rcos(\varphi) & -Rsin(\varphi)\\ -\frac{1}{2}Rcos(\varphi) & -\frac{\sqrt{3}}{2}Rcos(\varphi) & -Rsin(\varphi)\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}$$

traducendo invece le velocità angolari della sfera attorno agli assi x e y in velocità di traslazione della stessa si ottiene:

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \frac{1}{r}\cdot\begin{bmatrix} 0 & -cos(\varphi) & -Rsin(\varphi)\\ \frac{\sqrt{3}}{2}cos(\varphi) & \frac{1}{2}cos(\varphi) & -Rsin(\varphi)\\ -\frac{\sqrt{3}}{2}cos(\varphi) & \frac{1}{2}cos(\varphi) & -Rsin(\varphi)\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ \omega_z\end{bmatrix}$$

Si noti come la colonna 1 e 2 della matrice risultino invertite, inoltre la colonna relativa alla ω_x viene anche cambiata di segno poiché una rotazione attorno all'asse x produce una traslazione negativa lungo l'asse y.

Il comando di attuazione da imporre a ciascuna delle ruote dunque, risulta essere funzione della velocità della sfera che si vuole ottenere e dell'angolo con cui vengono montate le ruote del ballbot; la velocità della sfera è calcolata tramite il controllo del robot ed è univoca, l'angolo dipende dal punto di contatto della ruota con la sfera e sulla base delle considerazioni viste in precedenza, esso cambia nel caso in cui si utilizzino ruote omnidirezionali a due file di rullini.

4.2 Modello cinematico accurato

Supponendo l'utilizzo di ruote omnidirezionali a due file di rulli, in riferimento alla Figura 15, risultano due punti di contatto per ogni ruota, e di conseguenza 2 angoli φ per ogni ruota.

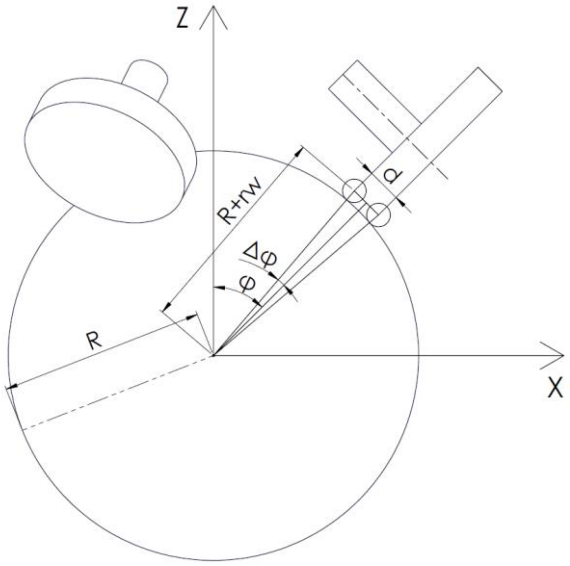


Figura 15: Contatto ruota omnidirezionale a 2 file di rulli

Come si nota nella Figura 15, l'angolo di contatto della ruota, utilizzato nell'analisi della matrice jacobiana fatta nel capitolo precedente, cambia di un fattore uguale a $\pm\Delta\varphi$ in base alla fila di rullini che è a contatto con la sfera in ogni istante.

Risulta:

$$\sin(\Delta\varphi) = \frac{d}{2 \cdot (R + r_w)}$$

$$\cos(\Delta\varphi) = \frac{1}{2 \cdot (R + r_w)} \cdot \sqrt{4 \cdot (R + r_w)^2 - d^2}$$

inoltre:

$$\sin(\varphi \pm \Delta\varphi) = \frac{1}{2 \cdot (R + r_w)} \cdot \left(\sqrt{4 \cdot (R + r_w)^2 - d^2} \cdot sin(\varphi) \pm d \cdot cos(\varphi)\right)$$

$$\cos(\varphi \pm \Delta\varphi) = \frac{1}{2 \cdot (R + r_w)} \cdot \left(\sqrt{4 \cdot (R + r_w)^2 - d^2} \cdot cos(\varphi) \mp d \cdot sin(\varphi)\right)$$

avendo 2 angoli possibili per ognuna delle 3 ruote esistono 2^3 combinazioni possibili di punti di contatto, pertanto esistono 8 diverse matrici jacobiane.

Volendo valutare l'errore che si commette approssimando il contatto all'angolo φ sono state fatte alcune prove numeriche fornendo in input le velocità angolari delle 3 ruote. Si è dapprima simulato il comportamento cinematico con le ruote in fase, ovvero lo stesso punto di contatto iniziale. In seguito si è introdotto uno sfasamento iniziale delle ruote. Nelle prove si considerato l'utilizzo di una ruota omnidirezionale avente 9 rullini per fila uguale a quella scelta per il prototipo.

4.3 Simulazioni mediante modello cinematico approssimato e accurata

Prova 1: Traslazione in direzione X

Velocità angolare w_1 [rad/s]	0
Velocità angolare w_2 [rad/s]	1
Velocità angolare w_3 [rad/s]	-1
Sfasamento iniziale	NO

Si è valutato l'errore commesso in termini di velocità evidenziando la differenza tra il modello cinematico approssimato e accurato; in

Figura 16 sono rappresentate le velocità tangenziali.

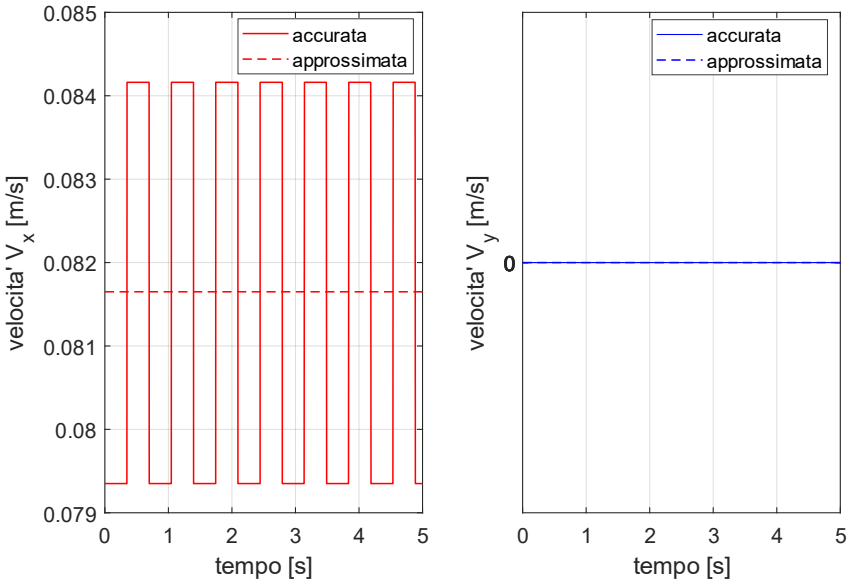


Figura 16: Velocità tangenziale nei due modelli cinematici. Simulazione traslazione in direzione X senza sfasamento iniziale

In questo caso non è stato introdotto alcuno sfasamento iniziale, ne consegue che essendo uguali le velocità delle due ruote attive, il punto di contatto si sposta con la stessa frequenza.

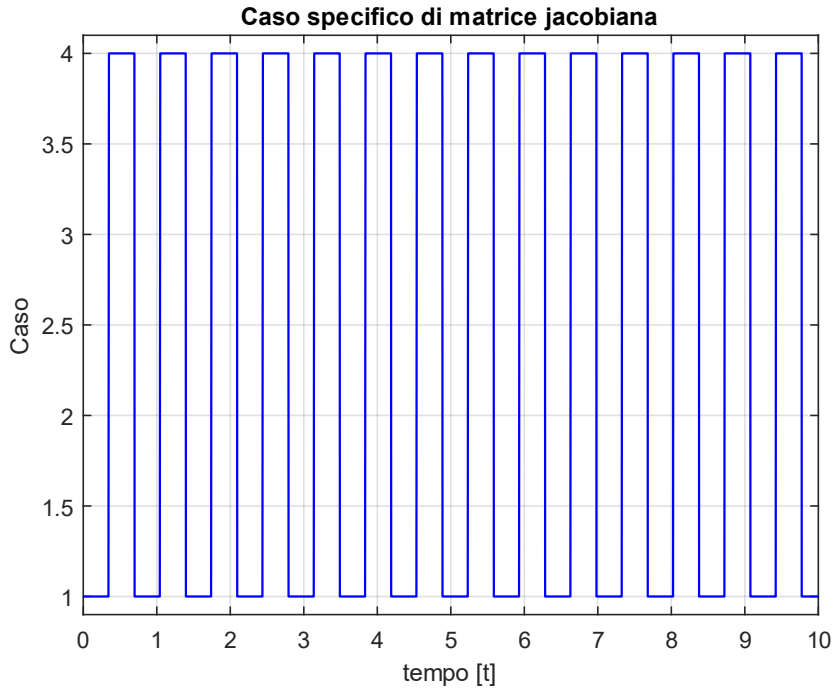


Figura 17: Caso che determina la matrice Jacobiana. Simulazione traslazione in direzione X senza sfasamento iniziale

In Figura 17 è mostrato come il caso di configurazione passa da 1, che corrisponde alla configurazione delle 3 ruote con rullino esterno in contatto, al caso 4, che corrisponde a ruota 1 con rullino esterno a contatto e ruote 2 e 3 con rullino interno a contatto. Questo è congruente con il grafico della velocità che risulta leggermente inferiore nel caso 1. Infatti la velocità è inversamente proporzionale all'angolo di contatto delle ruote.

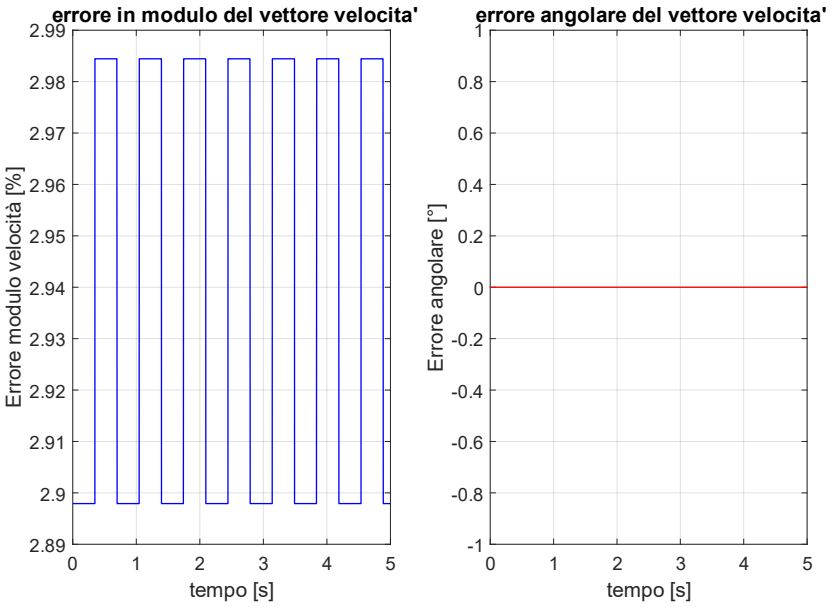


Figura 18: Errore commesso sulla velocità in termini di modulo e direzione. Simulazione traslazione in direzione X senza sfasamento iniziale

In Figura 18 è rappresentato l'errore angolare, e l'errore percentuale sul modulo della velocità calcolato come segue:

$$e_{\%} = \frac{|\|\vec{w}\| - \|\overrightarrow{w_0}\||}{\|\vec{w}\|} \cdot 100$$

Dove $\vec{w}$ è la velocità angolare della sfera calcolata con il modello accurato e $\overrightarrow{w_0}$ è la velocità angolare calcolata con il modello cinematico approssimato.

La velocità cambia modulo in funzione del punto di contatto come già analizzato in precedenza, ma non subisce errore in termini di direzione, questo è garantito dalla posizione iniziale delle 3 ruote che sono in fase. Pertanto anche la posizione riferita al sistema di riferimento globale non subisce errore.

Prova 2: Traslazione in direzione X

Velocità angolare w_1 [rad/s]	0
Velocità angolare w_2 [rad/s]	1
Velocità angolare w_3 [rad/s]	-1
Sfasamento iniziale	SI

Si effettua la medesima simulazione introducendo uno sfasamento iniziale che simuli la posizione casuale delle 3 ruote senza che avvenga un azzeramento. È subito visibile, analizzando il grafico in Figura 19, che in questo caso nasce una velocità in direzione y non voluta.

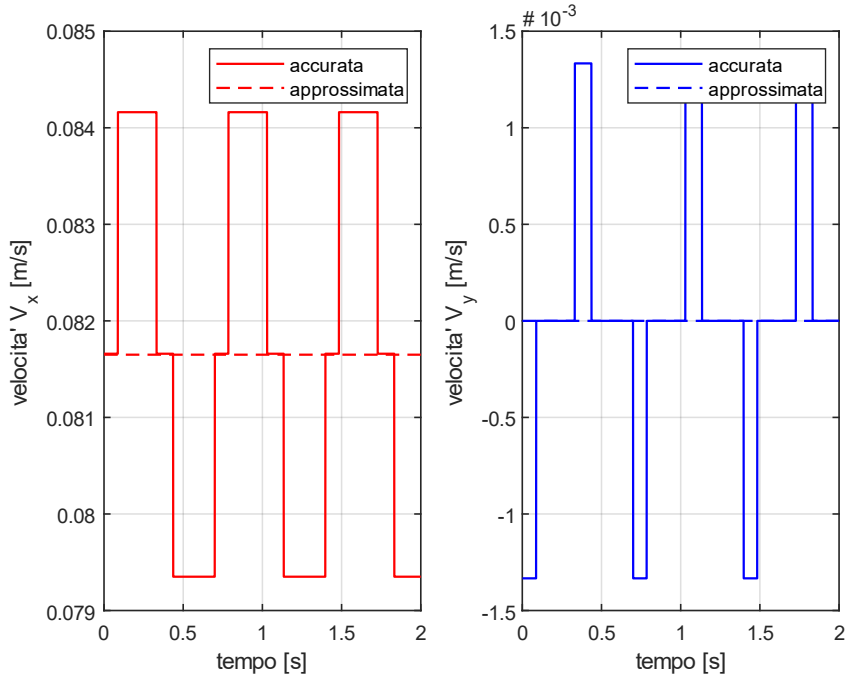


Figura 19: Velocità tangenziale nei due modelli cinematici.
Simulazione traslazione in direzione X con sfasamento iniziale

Lo sfasamento iniziale introdotto, fa si che il punto di contatto delle ruote attive, in questo caso 2 e 3, non cambi contemporaneamente, pertanto si susseguono tutte e 4 le configurazioni (Figura 20 in basso) relative alle due ruote, che definiscono la matrice Jacobiana.

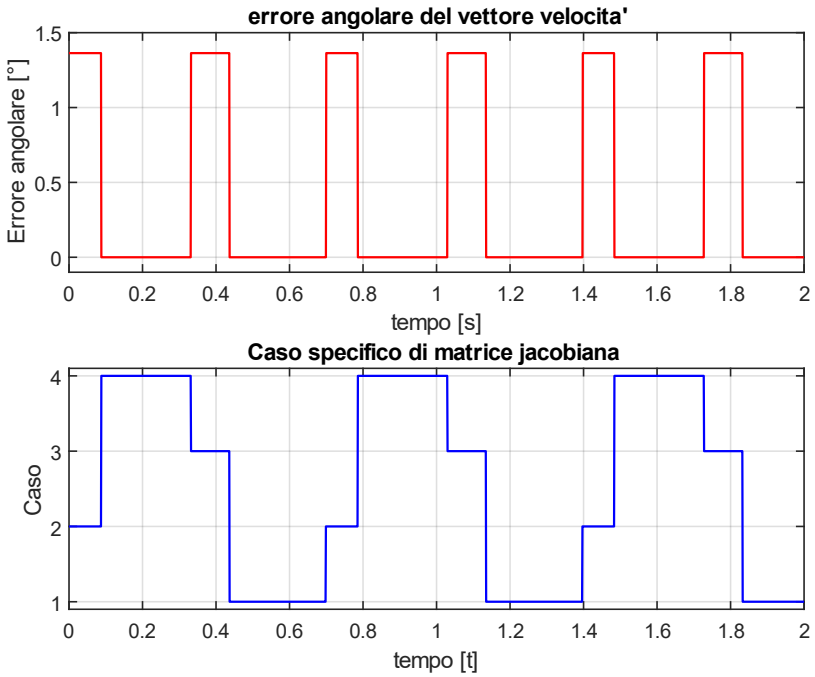


Figura 20: Caso che determina la matrice Jacobiana. Simulazione traslazione in direzione X con sfasamento iniziale

In questa simulazione, a differenza di quanto avveniva nel caso senza sfasamento, durante le configurazioni 2 e 3 di contatto, nasce un errore angolare alla velocità come visibile in Figura 20. Di conseguenza nasce una rotazione attorno all'asse Z del robot che produce un effetto di deriva del robot durante la traslazione.

Analizzando la posizione del robot nel sistema di riferimento globale in Figura 21, è visibile l'effetto della velocità in direzione y che introduce un'oscillazione e l'effetto dell'imbardata introdotta dalla rotazione attorno all'asse Z. Al fine di ottenere un risultato apprezzabile si è effettuata una simulazione con gli stessi parametri ma con durata 20 secondi. L'errore è apprezzabile ma comunque contenuto.

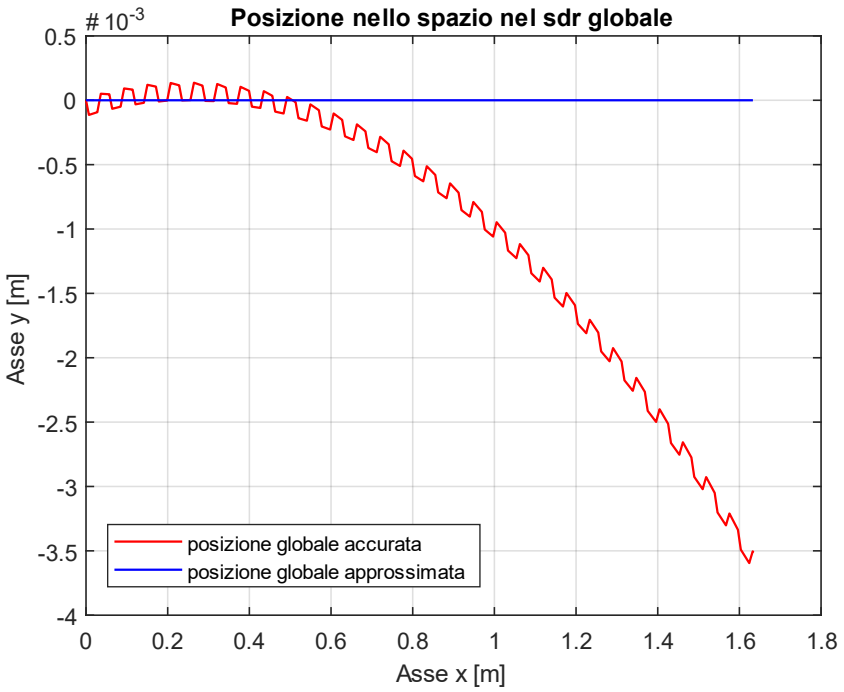


Figura 21: Posizione nel sistema di riferimento globale. Simulazione traslazione in direzione X con sfasamento iniziale

Prova 3: Traslazione in direzione Y

Velocità angolare w_1 [rad/s]	-1
Velocità angolare w_2 [rad/s]	0.5
Velocità angolare w_3 [rad/s]	0.5
Sfasamento iniziale	NO

In questa prova viene analizzato l'errore commesso durante una traslazione in direzione Y.

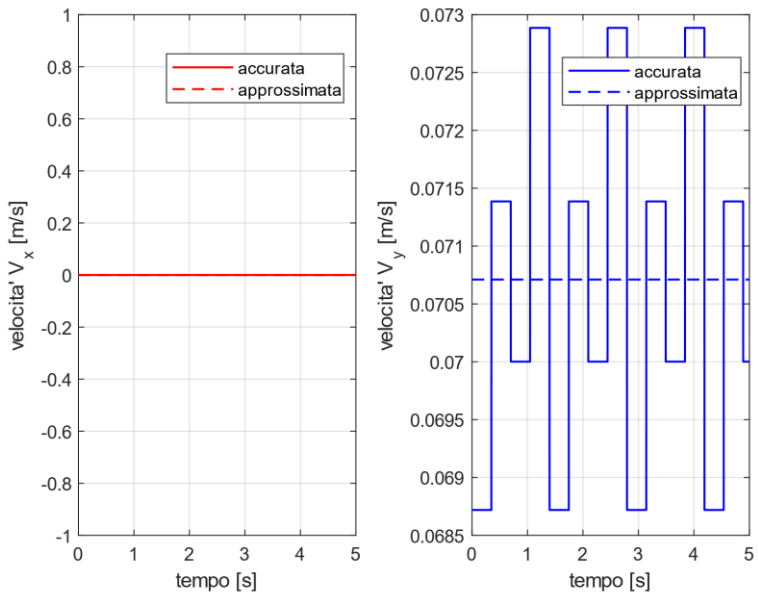


Figura 22: Velocità tangenziale nei due modelli cinematici
simulazione traslazione in direzione Y senza sfasamento iniziale

In questo caso, anche senza introdurre sfasamento, si genera un errore angolare (Figura 23) del vettore velocità che tuttavia non produce una velocità in direzione X rispetto al sdr solidale al robot quello che verrà poi definito R_0 nel modello simulink

Questo significa che l'errore angolare è un errore che produce rotazione attorno all'asse Z del sistema di riferimento globale e non del sistema di riferimento robot.

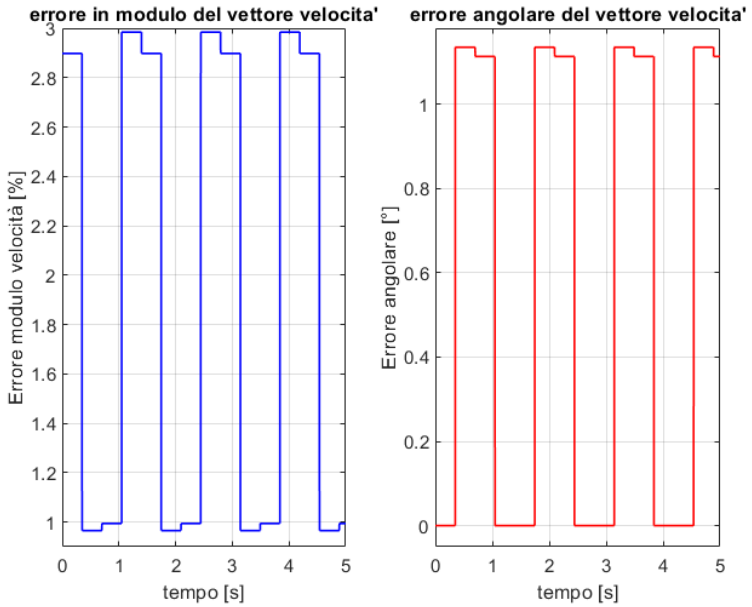


Figura 23: Errore commesso sulla velocità in termini di modulo e direzione. Simulazione traslazione in direzione Y senza sfasamento iniziale

Analizzando l'errore nei due sistemi di riferimento, si può apprezzare quanto descritto finora.

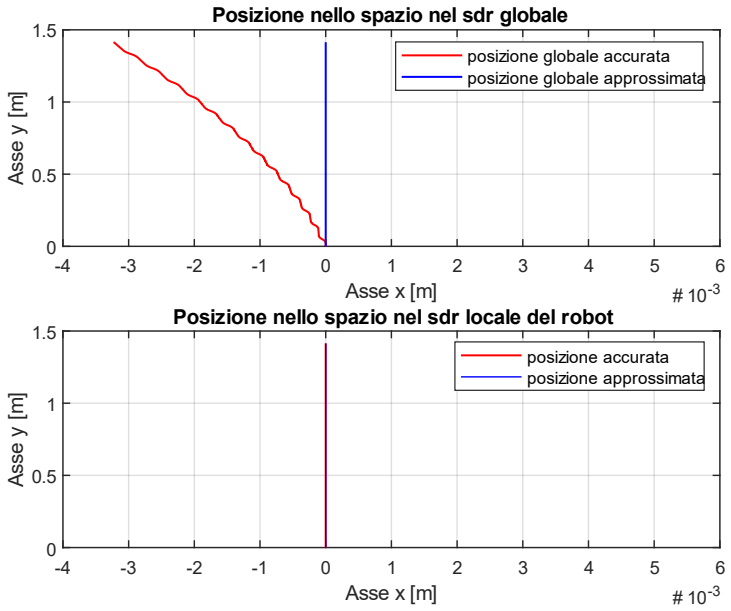


Figura 24: Confronto di lettura della posizione nei due sistemi di riferimento

Prova 3: Traslazione in direzione Y

Velocità angolare w_1 [rad/s]	-1
Velocità angolare w_2 [rad/s]	0.5
Velocità angolare w_3 [rad/s]	0.5
Sfasamento iniziale	SI

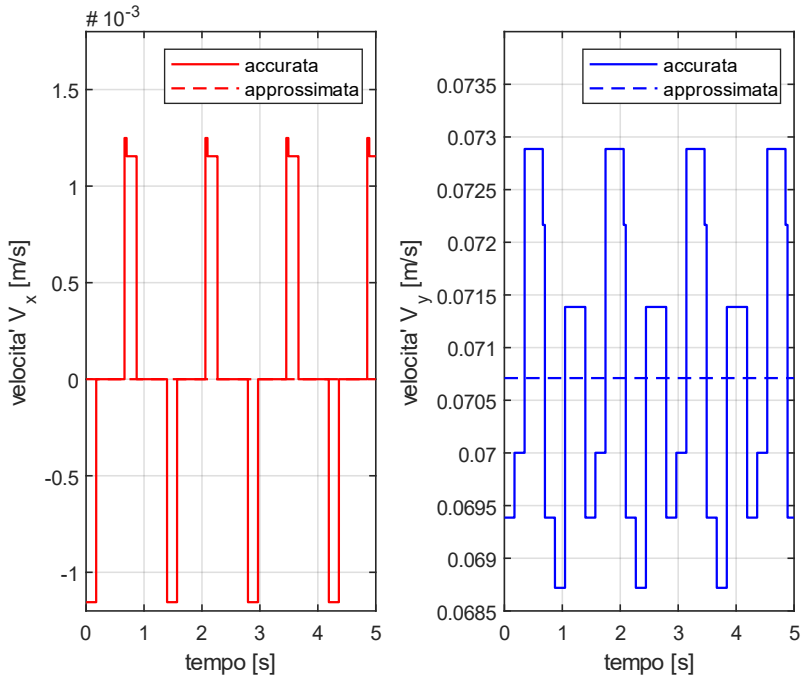


Figura 25: Velocità tangenziale nei due modelli cinematici simulazione traslazione in direzione Y con sfasamento iniziale

In questo caso lo sfasamento iniziale produce una componente di velocità in direzione X visibile in Figura 25.

Dalle prove effettuate simulando il modello cinematico, è emerso che l'errore commesso in termini di posizione è dell'ordine dei millimetri per simulazioni a basse velocità e per tempi contenuti. Aumentando le velocità di rotazione delle ruote e valutando l'effetto su simulazioni più durature nel tempo si apprezzerebbero errori più rilevanti, dell'ordine dei decimetri. Tuttavia si è deciso, almeno in un primo momento, di trascurare l'errore introdotto con l'utilizzo di ruote omnidirezionali a 2 file di rulli, per non appesantire il software di controllo. In un secondo momento, sarebbe possibile eliminare tale errore, sostituendo le ruote con ruote omnidirezionali a singola fila di rulli. In questo modo si eliminerebbero le combinazioni di configurazione e la cinematica sarebbe interamente descritta da una singola matrice Jacobiana.

5 MODELLO SIMULINK 3D

In questa sezione è descritto il modello 3D implementato su simulink sfruttando le librerie di simscape con le quali è possibile rappresentare sistemi multibody legando i sistemi di riferimento dei corpi mediante relazioni dinamiche o cinematiche. Il software è in grado di risolvere la dinamica dei gradi di libertà del modello. Un modello di simulazione di questo tipo è molto utile per poter valutare alcuni parametri fondamentali per il corretto funzionamento del prototipo. È stato indispensabile per individuare, mediante alcune simulazioni, le criticità che si sarebbero potute riscontrare nel prototipo. Inoltre, prima di acquistare i componenti necessari alla realizzazione del ballbot, sono stati dimensionati i vari componenti, verificando sul modello le prestazioni finali del robot.

Il modello è sviluppato in sotto-blocchi ognuno dei quali svolge una funzione precisa.

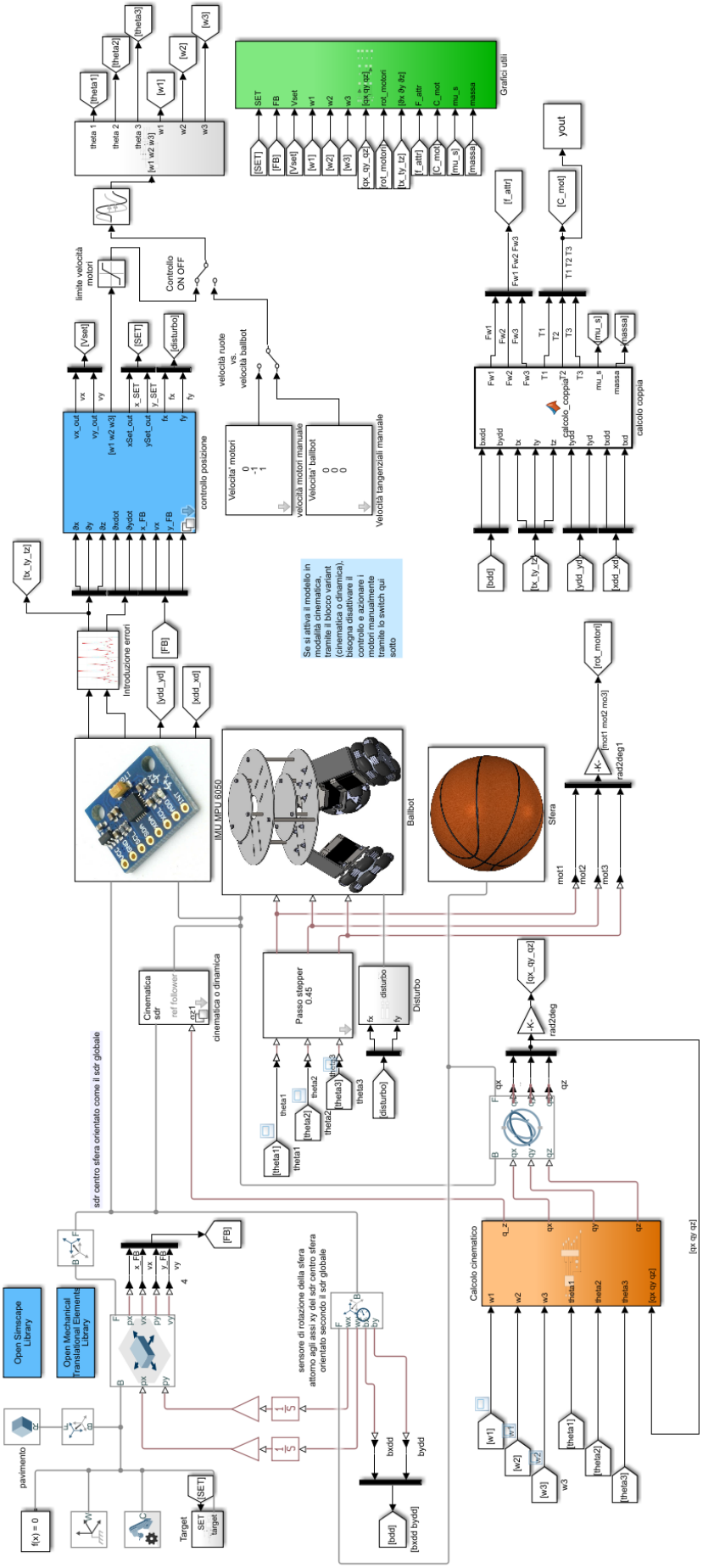


Figura 26: Modello simulink

Al fine di inserire nel modello le relazioni tra i corpi, sono stati introdotti alcuni sistemi di riferimento essenziali per identificare le parti e per descriverne i gradi di libertà.

Sistema di riferimento	origine	Orientazione
World frame	-	-
S0	Centro sfera	Come il World frame
R0	Centro sfera	Solidale con il robot. Asse Z coincidente con l'asse delle piastre e Asse X coincidente in direzione del motore 1.
S1	Centro sfera	Solidale alla sfera

Le ipotesi sotto le quali è stato creato il modello, analogamente a quanto fatto per il modello semplificato 2D, sono condizioni di rotolamento puro tra sfera e terreno e tra ruote e sfera. In questo modo, le relazioni tra ruote e sfera, e tra sfera e terreno, possono essere descritte da relazioni cinematiche lasciando come incognita al software la risoluzione della dinamica.

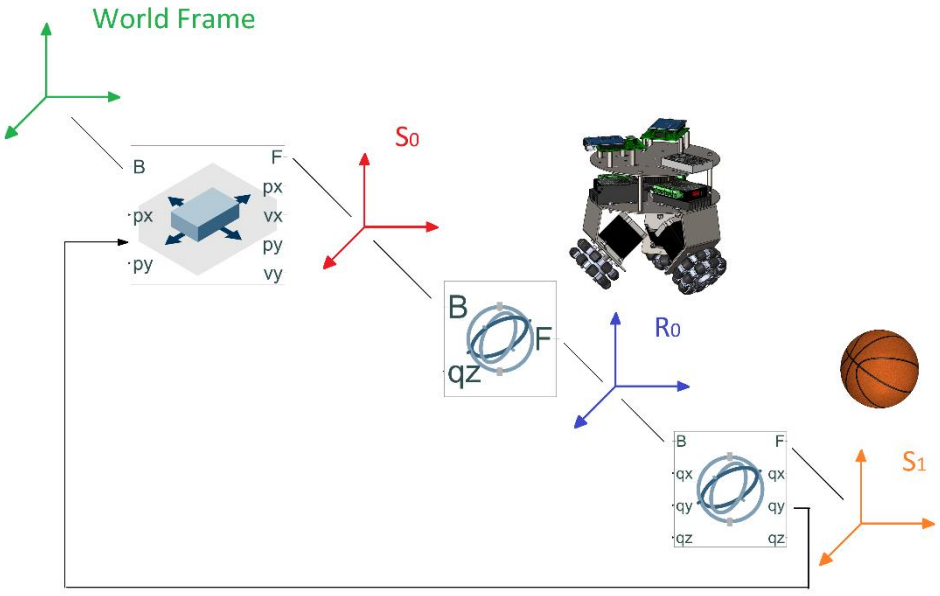


Figura 27: Schema logico delle relazioni che intercorrono tra i sistemi di riferimento

In Figura 27 è mostrata la logica con cui sono stati messi in relazione i sistemi di riferimento all'interno del modello. Le lettere B e F presenti a sinistra e destra di ogni blocco relazionale indicano la gerarchia del giunto. Lettera B, che sta per "Base", è il sistema di riferimento fisso; invece F, che sta per "Follower", è il sdr mobile. Partendo dall'ultimo grado gerarchico, il sistema di riferimento S1, è legato al giunto R0 mediante un giunto cardanico, in input sono fornite le rotazioni attorno ai tre assi X, Y e Z nel seguente ordine. R0 è a sua volta follower di S0 in un secondo giunto cardanico. A differenza del primo giunto, quest'ultimo non presenta in input le rotazioni attorno a X e Y ma solo attorno a Z. S0, follower di un giunto di traslazione, trasla rispetto al world frame secondo la cinematica della sfera, la soluzione dell'orientazione di R0 è affidata al software il quale dopo aver risolto la dinamica, impone le rotazioni a R0. In questo modo, essendo una risoluzione nello spazio, le relazioni dinamiche tra i due piani ortogonali, che non erano state considerate nel modello 2D, sono considerate durante la risoluzione.

Di seguito sono analizzati i blocchi presenti nel modello.

5.1 Blocco ballbot

Al blocco vengo forniti tre input fisici che rappresentano le rotazioni delle ruote omnidirezionali. Le altre due linee (in grigio) rappresentano i sistemi di riferimento.

Nel blocco sono definiti i sistemi di riferimento dei corpi che compongono il robot e le relazioni che intercorrono tra essi.

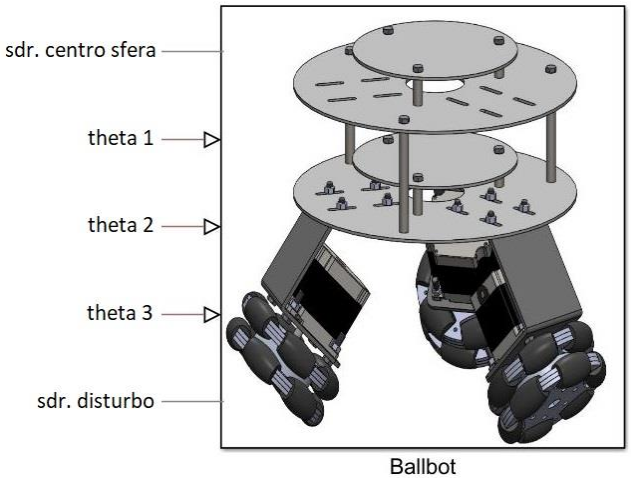


Figura 28: Blocco ballbot

Ogni corpo viene caricato tramite file .STEP generato dal software SolidWorks; tale file comprende il modello 3D della parte e il sistema di riferimento. La matrice di inerzia viene calcolata automaticamente dalla geometria.

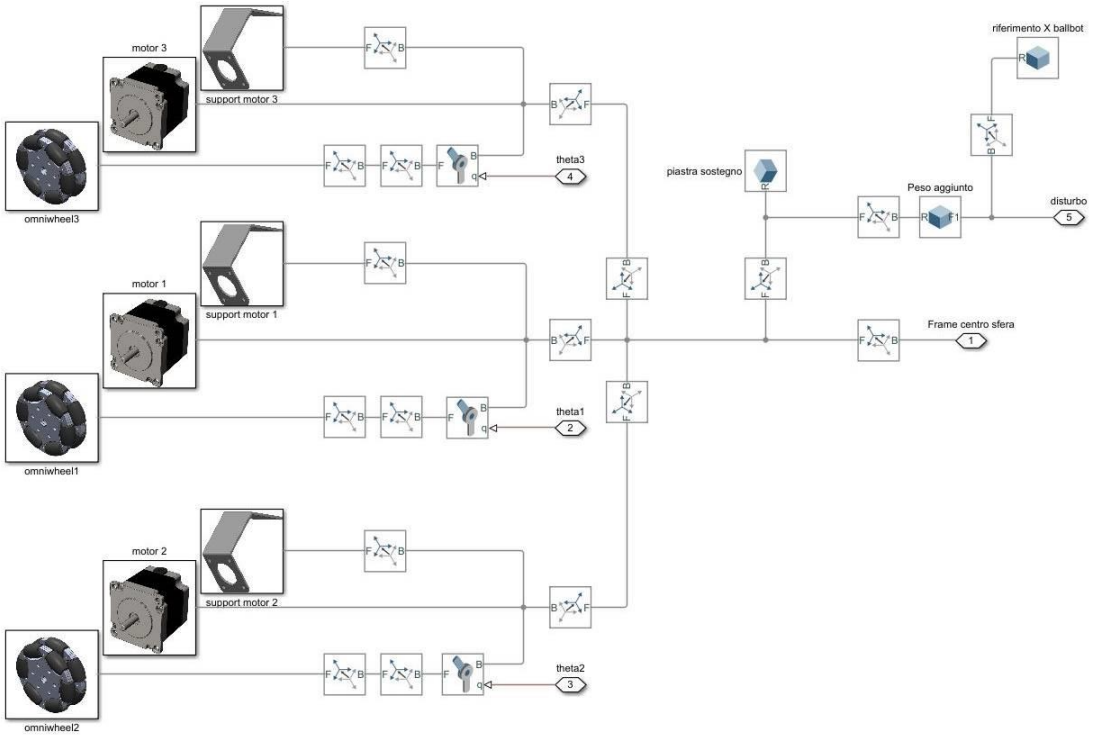


Figura 29: Blocco ballbot esteso

Nella Figura 29 è mostrato il blocco ballbot per esteso. Ogni sistema di riferimento del relativo file .STEP è legato a un sistema di riferimento tramite una relazione fissa di rotazione e traslazione. In questo modo tutti i corpi solidi presenti in questo blocco costituiscono un unico corpo rigido, a meno delle ruote le quali sono legate ai motori tramite un giunto rotoidale che presenta un singolo grado di libertà di rotazione attorno all'asse z; tale grado di libertà viene fornito in input al blocco.

È stato aggiunto il sistema di riferimento R_0 rappresentativo dell'intero corpo rigido avente origine nel centro della sfera, orientazione dell'asse Z secondo l'asse della piastra di sostegno e asse X orientato in direzione del motore 1. Tale sistema di riferimento è mostrato in azzurro in Figura 30 .

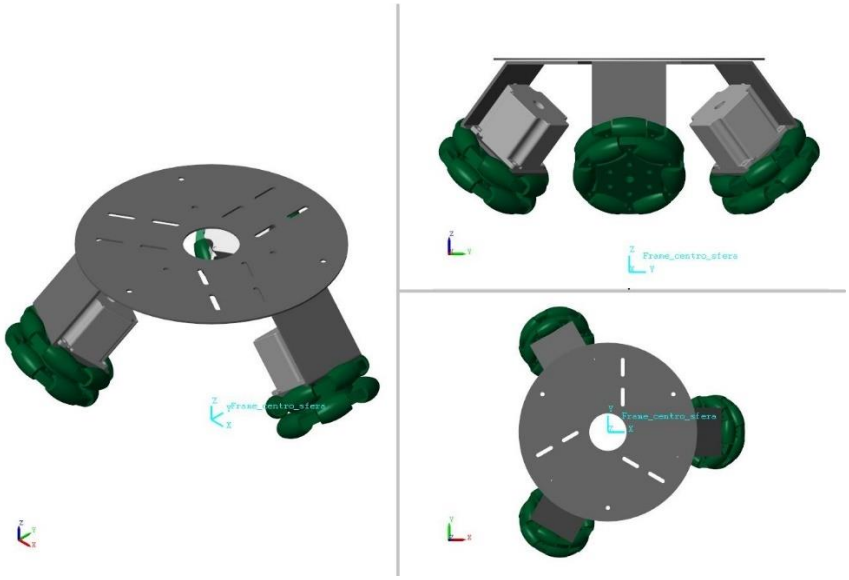


Figura 30: Sistema di riferimento centro sfera

Il robot presenta nello spazio 5 gradi di libertà, di cui 3 rotazioni attorno agli assi e 2 traslazioni dell'origine di R_0 lungo gli assi x e y del sistema di riferimento globale rappresentato in Figura 30 in basso a sinistra delle 3 immagini.

5.2 Blocco cinematica vs. dinamica

Il legame che intercorre tra il robot identificato da R_0 e S_1 è un legame di tipo sferico in quanto R_0 può orientarsi a piacere mantenendo le l'origine nel centro sfera. Per creare questo legame all'interno del modello viene creato un ulteriore sistema di riferimento; S_0. Esso ha origine nel centro sfera e orientazione uguale al sistema di riferimento globale.

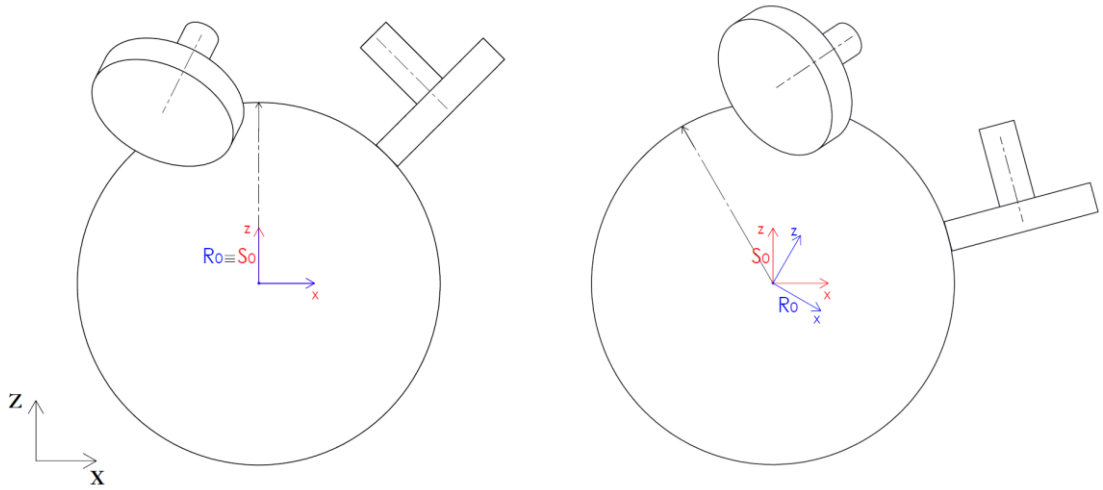


Figura 31: Sistemi di riferimento R0 e S0 nel piano xz

In questo modo è possibile legare R_0 a S_0 tramite un giunto cardanico il quale permette tre rotazioni successive attorno rispettivamente agli assi xyz del sistema di riferimento mobile (Follower). Il modello permette di scegliere due modalità di simulazione, una dinamica e una cinematica. Nel secondo caso il giunto cardanico è sostituito da un giunto rotoidale eliminando 2 gradi di libertà del robot il quale non può "sbilanciarsi" ma solo traslare lungo le direzioni x e y e ruotare attorno all'asse z. La dinamica del giunto cardanico R_0-S_0, eccetto la rotazione di R_0 attorno al proprio asse z, che viene fornita in input al blocco, viene interamente risolta dal software. Si è deciso di considerare l'attrito tra sfera e terreno sufficiente a non permettere la rotazione di S_1 attorno all'asse z di S_0. Pertanto la componente verticale di velocità angolare complessiva delle tre ruote viene imputata interamente alla rotazione di R_0. In Figura 31 è presente una freccia di riferimento fissa alla sfera, per indicare che può esserci una rotazione

relativa tra il sistema di riferimento R_0 e il sistema di riferimento S_1. La relazione cinematica tra le ruote e la sfera impone l'orientazione di S_1 rispetto R_0, e la risoluzione della dinamica del sistema definisce la rotazione di R_0 attorno a S_0.

5.3 Blocco cinematica sfera

Nel paragrafo precedente è stato descritto il legame che intercorre tra il sdr del robot R_0 e il centro della sfera S_0, nel paragrafo seguente si definisce la relazione tra la sfera S_1 e R_0. Anche in questo caso l'origine degli assi è in comune ai due sistemi di riferimento e S_1 ha tre gradi di libertà che rappresentano le tre rotazioni attorno ai tre assi, pertanto si tratta di un accoppiamento sferico. Analogamente a quanto visto in precedenza si rappresenta mediante un giunto cardanico. In questo caso però la rotazione della sfera è univocamente definita dalla rotazione delle tre ruote sotto l'ipotesi di rotolamento puro. Dalla risoluzione della cinematica del robot, essendo i tre motori solidali al sistema di riferimento R_0, si ricavano le velocità di rotazione della sfera attorno agli assi di R_0 x e y; definiamo tale velocità ${}^{R_0}\Omega$ per indicare che si tratta di velocità angolari riferite agli assi del sistema di riferimento R_0. Tuttavia il giunto cardanico richiede in input 3 rotazioni successive attorno agli assi x y e z del sistema di riferimento mobile S_1., è necessario pertanto riferire la velocità angolare al sistema di riferimento S_1.

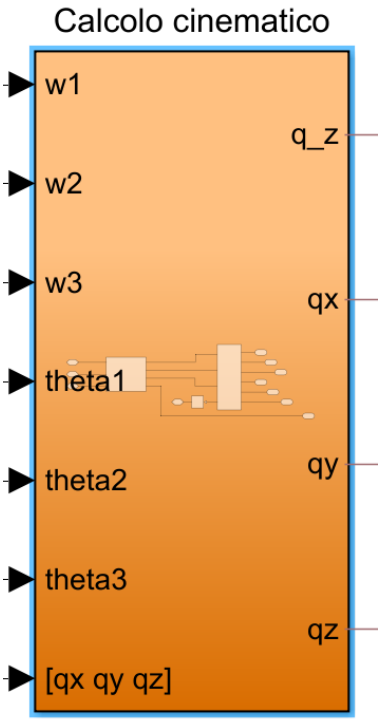


Figura 32: Calcolo cinematico

La risoluzione di questo problema avviene all'interno del blocco *Calcolo cinematico* rappresentato a fianco in Figura 32. Sono forniti in input la velocità e la rotazione delle tre ruote rispettivamente w_1 w_2 w_3 e $theta_1$ $theta_2$ $theta_3$, $[q_x, q_y q_z]$ rappresentano le rotazioni successive xyz attorno agli assi mobili per portare R_0 a coincidere con S_1 nell'istante considerato.

La relazione che intercorre tra R_0 e S_1 in ogni istante è rappresentata da questa relazione:

$$^{R_0}A_{S_1} = rot(x, q_x) \times rot(y, q_y) \times rot(z, q_z)$$

Le tre rotazioni successive eseguite in post-moltiplicazione indicano la rotazione attorno all'asse del sistema di riferimento in seguito alla rotazione.

Per quanto riguarda la velocità $^{R_0}\Omega$, ovvero la velocità angolare rispetto al sistema di riferimento R_0, è necessario trasformarla secondo la convenzione utilizzata dal blocco del giunto cardanico presente nel modello, ovvero una convenzione $RPY_{[xyz]body}$. Risulta pertanto:

$$^{R_0}\Omega = \dot{\psi} \cdot i_0 + \dot{\vartheta} \cdot j_1 + \dot{\varphi} \cdot k_2$$

$\dot{\psi}, \dot{\vartheta}\ e\ \dot{\varphi}$ sono le velocità angolari secondo al convenzione utilizzata nel modello, infatti:

$$\mathrm{i}_0\text{: la prima colonna di } {}^{\mathrm{R}_0}\mathrm{A}_{\mathrm{R}_0},$$

ovvero l'asse x di R_0

$$\mathrm{j}_1\text{: la seconda colonna di } {}^{\mathrm{R}_0}\mathrm{A}_{\mathrm{R}_0} \times rot(x, q_x),$$

ovvero l'asse y di R_0 dopo aver subito la prima rotazione attorno ad x

$$k_2\text{: la terza colonna di } {}^{\mathrm{R}_0}\mathrm{A}_{\mathrm{R}_0} \times rot(x, q_x) \times rot(y, q_y),$$

ovvero l'asse z di R_0 dopo aver subito la prima rotazione attorno ad x e la successiva rotazione attorno a y del sdr ruotato.

Risolvendo l'equazione precedente si ricava:

$$^{R_0}\Omega = \begin{bmatrix} 1 & 0 & sin(q_y) \\ 0 & cos(q_x) & -sin(q_x)cos(q_y) \\ 0 & sin(q_x) & cos(q_x)cos(q_y) \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ \dot{\vartheta} \\ \dot{\varphi} \end{bmatrix}$$

Eseguendo l'inversa della matrice e pre-moltiplicandola a destra e sinistra dell'uguale, è possibile ricavare le velocità angolari riferite al sistema di riferimento mobile S_1; integrando le velocità angolari così ricavate si ottengono i valori q_x q_y q_z da fornire in input al giunto cardanico.

Gli output del blocco in esame sono:

q_z che è la rotazione di R0 attorno all'asse z che interviene nel giunto cardanico descritto nel paragrafo precedente.

q_x, q_y e q_z che rappresentano le tre rotazioni da fornire in input al giunto cardanico R_0-S_1 ricavate come appena descritto.

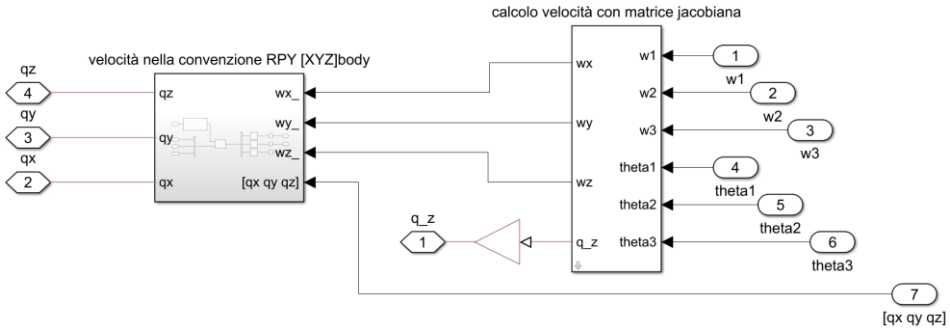


Figura 33: Blocco calcolo cinematico esteso

In Figura 33 è mostrato il blocco per esteso; nel blocco sulla sinistra, mostrato per esteso in Figura 34, viene eseguita la parte appena descritta di trasformazione della velocità angolare $^{R_0}\Omega$ nella convenzione utilizzata nel modello.

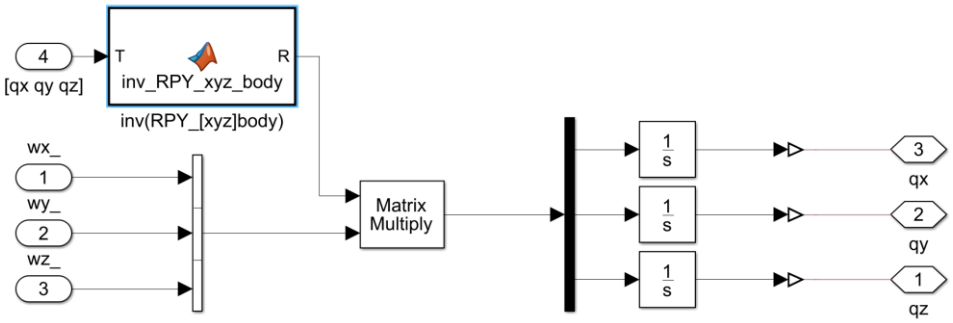


Figura 34: Velocità nella convenzione RPY[xyz]body

Nel blocco di destra di Figura 33 rappresentato per esteso in Figura 35 viene eseguito il calcolo di cinematica accurata descritto nel capitolo 3.

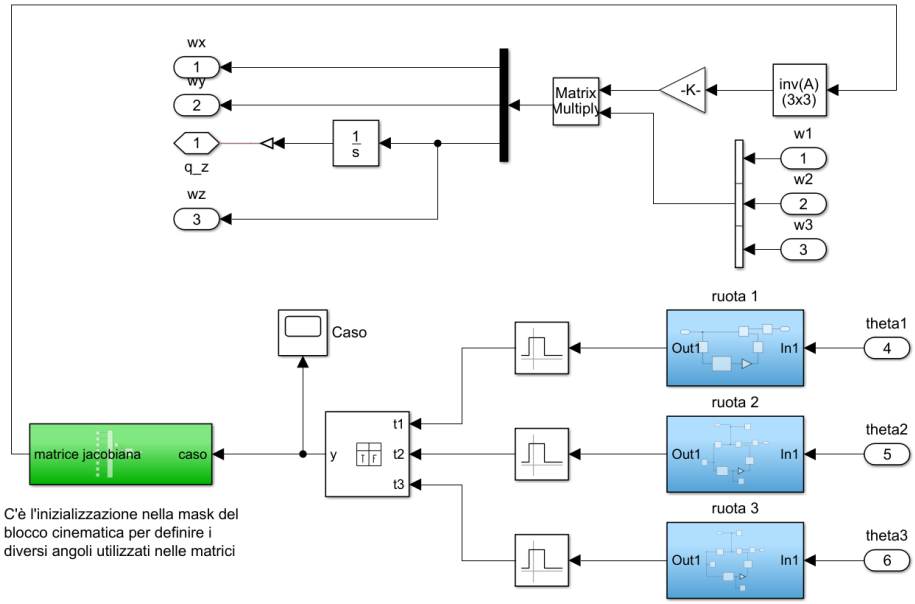


Figura 35: Blocco calcolo velocità con matrice jacobiana

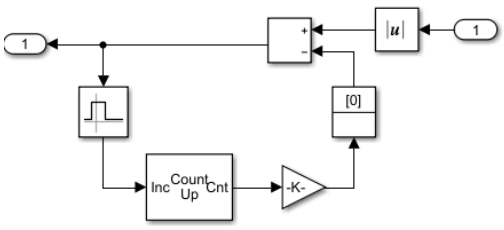


Figura 36: Generazione della funzione dente di sega

Gli input theta1 theta2 e theta3 sono gli angoli di rotazione delle ruote avendo già introdotto lo sfasamento iniziale; nei blocchi blu in basso a destra, rappresentati per esteso in Figura 36 viene trasformato l'angolo theta in una funzione a dente di sega di ampiezza $\frac{2\pi}{N}$ ovvero l'ampiezza

dell'angolo che contiene due rullini successivi. Tale funzione viene poi comparata nel blocco interval test con un valore costante pari a $\frac{2\pi}{2N}$ in questo modo, riferendoci ad esempio alla ruota 1, il segnale t1 assume il valore 1 quando è in appoggio il rullino interno e assume il valore 0 quando è in appoggio il rullino esterno.

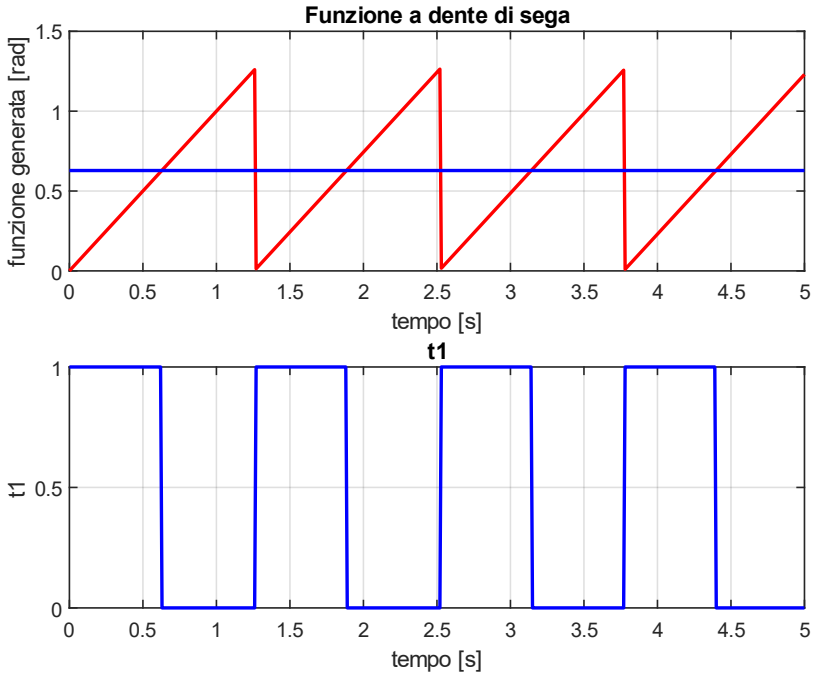


Figura 37: input booleano per la definizione della matrice jacobiana

I tre segnali t1, t2 e t3 così ricavati, sono gli input di una tabella della verità per la determinazione della matrice jacobiana. Le 2^3 combinazioni di t1, t2 e t3 equivalgono alle 8 possibili configurazioni di appoggio delle ruote omnidirezionali, pertanto è selezionata la matrice corrispondente alla configurazione attiva nell'istante considerato e si risolve la cinematica come descritto nel capitolo 2. Tale selezione avviene nel blocco verde di Figura 35 rappresentato in

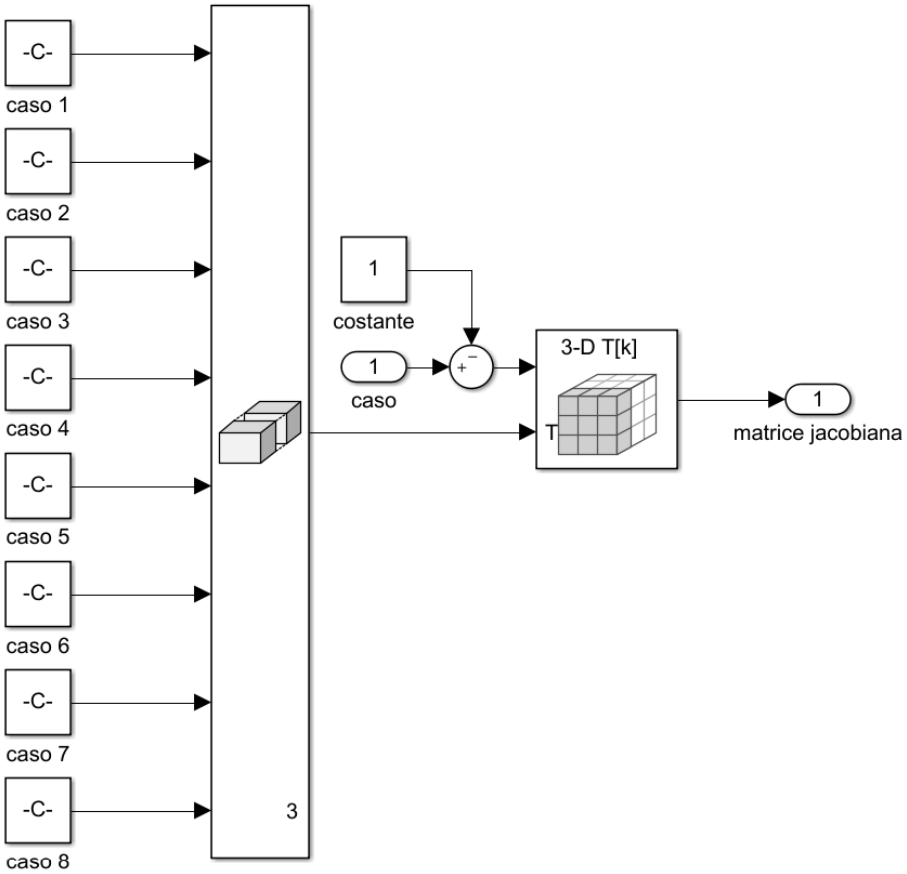


Figura 38: Struttura di matrici contenenti le 8 configurazioni di Jacobiana

5.4 Blocco controllo

Per quanto riguarda il controllo della stabilità del robot, all'interno del modello è stato implementato un algoritmo di controllo di tipo PID agente su due piani considerati indipendenti XZ e YZ, Il controllo cicla sulla posizione di R_0 nello spazio e sull'orientazione di R_0 rispetto S_0.

Nel modello è stato aggiunto un sensore che simuli una scheda IMU (inertia mesurament unit), in grado di leggere l'inclinazione del robot nei due sistemi di riferimento XZ e YZ con riferimento a R0. In questo modo è possibile calcolare le due accelerazioni $\vec{a}_x$ e $\vec{a}_y$, integrarle in velocità $\vec{v}_x$ e $\vec{v}_y$ e applicare il principio di sovrapposizione degli effetti. Risolvendo poi il problema cinematico trattato nel capitolo 2, si ricavano le velocità angolari delle ruote.

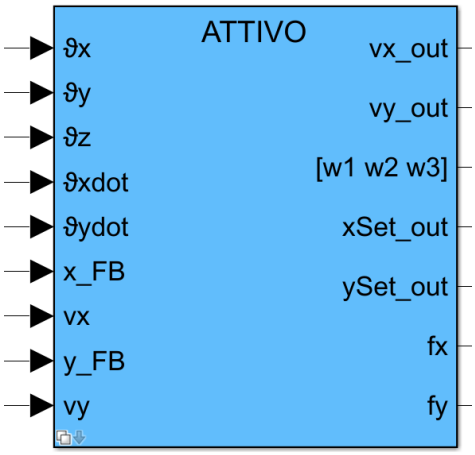


Figura 39: Blocco controllo

Quanto descritto avviene nel blocco Controllo posizione rappresentato in Figura 39. Si tratta di un blocco di tipo variant con la possibilità di scegliere tre diverse configurazioni. Queste tre configurazioni permettono di disinserire il controllo sulla posizione e mantenere unicamente il controllo sulla stabilità oppure di fornire in input la posizione x e y tramite equazione parametrica. Si analizza quanto avviene nel controllo attivo di posizione e orientazione nel piano XZ; nel piano YZ il ragionamento è analogo.

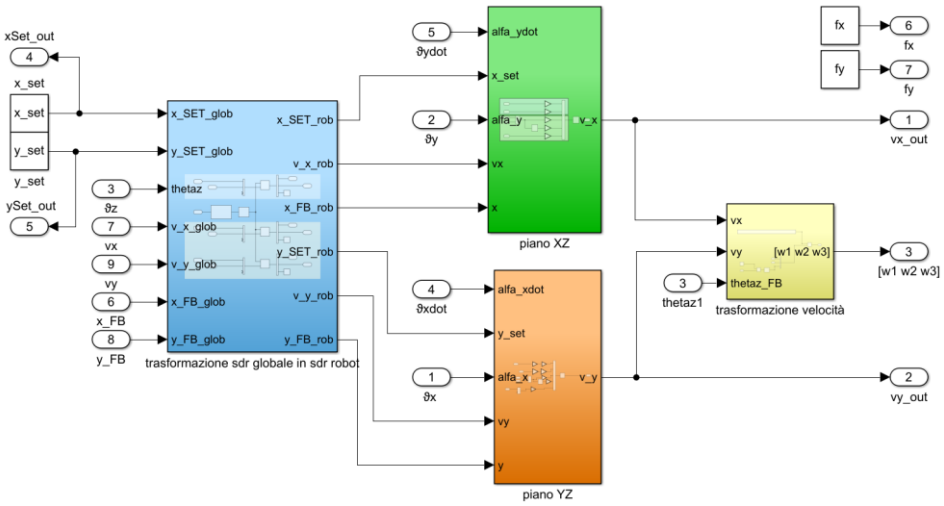


Figura 40: Blocco controllo

In Figura 40 è rappresentato il blocco controllo per esteso; sono forniti in input la posizione desiderata (x_set e y_set), la posizione attuale del ballbot (x_FB e y_FB), le componenti di velocità di R_0 in direzione x e y, l'orientazione di R_0 intesa come rotazione attorno agli assi x, y e z e le rispettive derivate, ovvero le velocità angolari.

Nel modello, le grandezze di feedback di posizione e velocità lineare, sono lette nel sistema di riferimento S_0, pertanto, per poter analizzare il problema nei due piani XZ e YZ di R_0, risulta necessario trasformare tali grandezze secondo quel sistema di riferimento, questo è quanto avviene nel blocco blu rappresentato in Figura 40. Per quanto riguarda invece le orientazioni e le velocità angolari, essendo la scheda IMU solidale al corpo robot, quindi solidale a R_0, le grandezze misurate sono già riferite al tale sistema di riferimento.

Per alleggerire il calcolo computazionale, si è scelto di trascurare la rotazione di R_0 attorno agli assi x e y considerando unicamente la rotazione attorno all'asse Z. Questa approssimazione è accettabile poiché è importante che il vettore di accelerazione generato dal controllo appartenga al piano analizzato, mentre sul modulo un piccolo errore non compromette il comportamento.

In Figura 41 è mostrata la logica di conversione dei parametri, in azzurro vi è la conversione dei parametri di SET e in verde la conversione dei parametri di FB. Nel blocco che richiama la funzione Matlab viene creata la matrice di rotazione della quantità in input considerando unicamente la rotazione attorno all'asse Z.

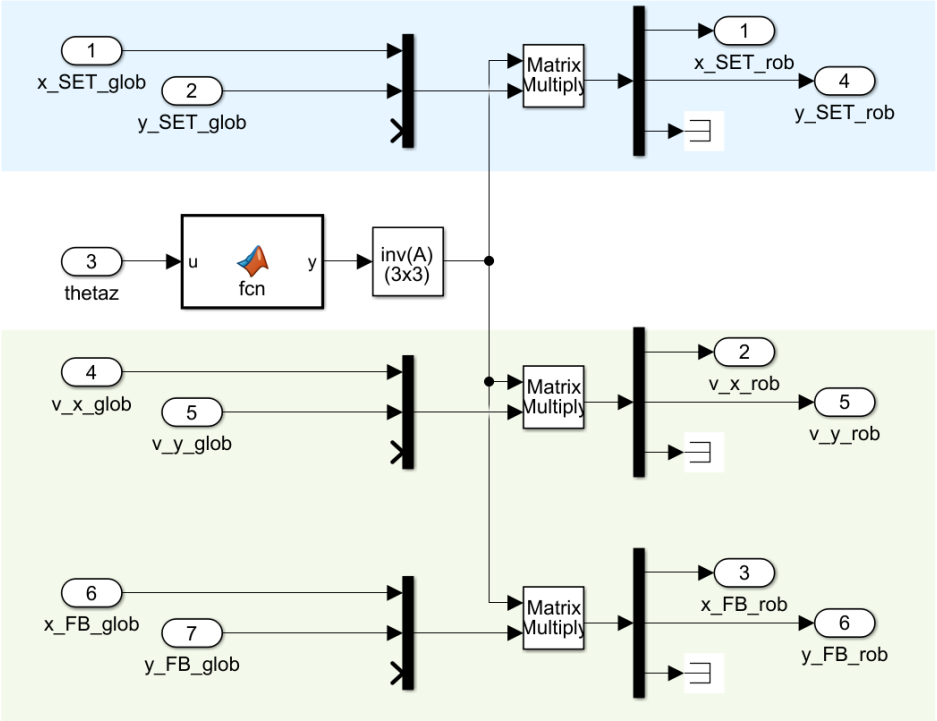


Figura 41: Blocco trasformazione sdr nel blocco controllo

L'equazione che regola tale conversione è la seguente:

$$^{R'_0}x = \left(^{S_0}A_{R'_0}\right)^{-1} \cdot {}^{S_0}x \ ^{3}$$

Le grandezze così trasformate possono essere elaborate nei blocchi di controllo PD.

[3] Si noti che nell'equazione si è scritto R'_0 per indicare l'approssimazione del sistema di riferimento considerando unicamente la rotazione attorno a z.

Per quanto riguarda la stabilità e la posizione del robot, il controllo è di tipo PD. La sezione azzurra rappresentata in Figura 42, è la parte di controllo dell'orientazione di R_0. Il set desiderato è 0, ovvero mantenere R0 orientato come S0 (in posizione verticale, senza sbilanciarsi). Per quanto riguarda il controllo sulla posizione, indicata in verde in Figura 42, è stata aggiunta una componente integrativa con lo scopo di ridurre l'errore a regime nel caso in cui il controllo sia sulla velocità, ovvero quando è fornito un input di set che cambia nel tempo secondo una legge parametrica. Imponendo il guadagno Ki pari a zero il controllo rimane di tipo PD.

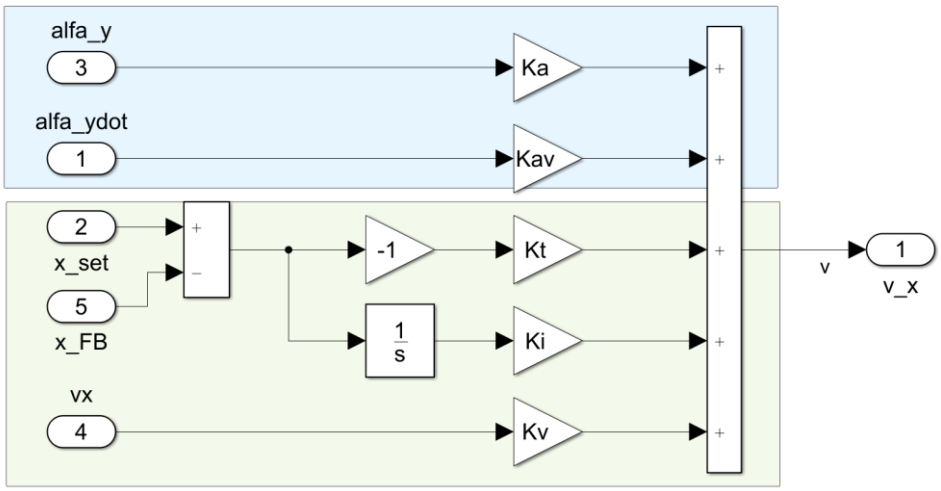


Figura 42: Controllo posizione e orientazione

La somma degli errori compensati determina l'accelerazione lineare da imporre alla sfera. Si noti come la componente proporzionale di errore relativa alla posizione, calcolata dalla formula $e = x_{SET} - x_{FB}$, in questo caso risulta cambiata di segno. Questo è dovuto alla logica con cui il robot si muove nello spazio. Per percorrere ad esempio una traslazione nello spazio in direzione +x la sfera deve subire un'accelerazione in direzione -x al fine di ottenere un'inclinazione del robot in direzione x (immagine 1 in Figura 43). Ottenuto così lo sbilanciamento iniziale, il controllo di orientazione agisce contro il controllo posizione, producendo una traslazione in direzione x in modo tale da recuperare lo sbilanciamento (immagine 2 in Figura 43). Avvicinandosi alla posizione desiderata, per poter decelerare, è

necessario imporre un'accelerazione per ottenere un'inclinazione in direzione -x (immagine 3 in Figura 43), compensata in fase di decelerazione.

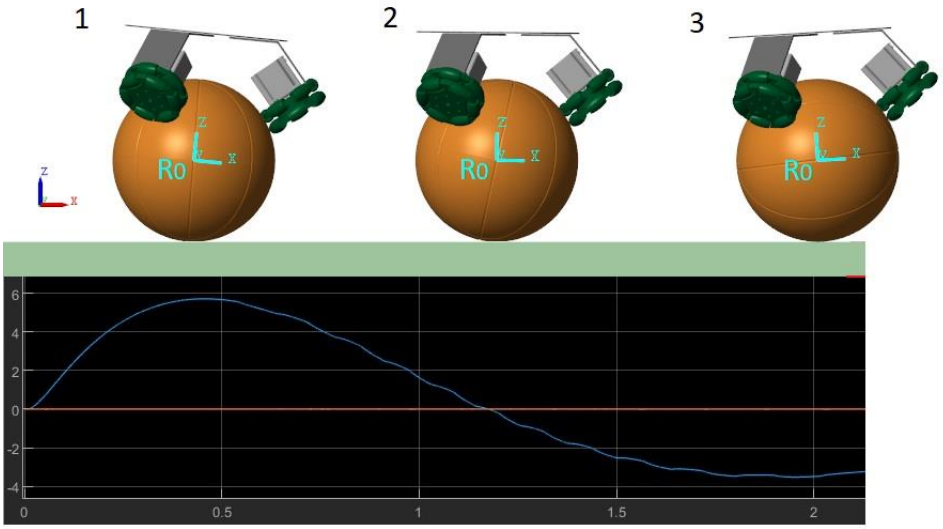


Figura 43: simulazione della traslazione in direzione x

In Figura 43 sono mostrati i tre fermi immagine corrispondenti alle tre fasi descritte. Sotto ogni fase è indicata l'inclinazione corrispondente in funzione del tempo.

6 DIMENSIONAMENTO PARTI

In questo capitolo si mostra il procedimento con il quale sono stati determinati alcuni parametri indispensabili per il dimensionamento dei componenti.

In particolare, si è sfruttato il modello dinamico più accurato implementato in ambiente simulink, tramite il quale è stato possibile effettuare diverse simulazioni, sia in termini di spostamento, ovvero applicando al modello degli input diversi, sia in termini di controllo, ovvero verificando il comportamento dinamico al variare dei guadagni del controllo a parità di set, sia dal punto di vista dinamico, ovvero variando il carico pagante in termini di peso o di equilibrio oppure introducendo un disturbo in termini di forza lineare.

Da queste simulazioni sono stati estratti i dati significativi e, applicando il modello semplificato 2D ricavato in precedenza, si sono ottenuti i dati relativi alle coppie motrici richieste ai motori e al contatto ruota sfera in termini di attrito.

6.1 Motori

Per quanto riguarda i motori, si è scelto di utilizzare i motori passo-passo, sia per la facilità di controllo mediante apposito driver, sia per la possibilità di avere prestazioni accettabili per il primo prototipo, il tutto ad un costo contenuto se paragonati ai motori brushless, i quali presentano prestazioni dinamiche più elevate ma richiedono un controllo più complesso.

Quindi, avendo definito la tipologia dei motori, le caratteristiche che devono avere i motori scelti dipendono dalle prestazioni minime richieste al ballbot.

Un parametro fondamentale nella scelta del motore è la caratteristica di coppia in particolare la curva di pull-out del motore passo passo. Mentre i servomotori brushless sono controllati in coppia o velocità, i motori stepper sono controllati in posizione. Fornendo al driver un treno di impulsi, ognuno dei quali corrisponde a una rotazione dell'albero motore di un passo, il driver modula l'intensità di corrente che percorre ciascun avvolgimento dello statore in modo tale da spostare il punto di equilibrio del rotore nella posizione desiderata. Tuttavia, il motore è in grado di seguire il passo imposto solo se la coppia resistente all'albero è inferiore alla coppia erogabile dal motore in quelle condizioni di funzionamento. Mappando la coppia erogabile per ogni velocità di rotazione dell'albero, quindi variando la frequenza portante del treno di impulsi forniti al driver, si ricava la curva di pull-out del motore che ne definisce la zona di funzionamento. Tale curva di coppia è ricavata sperimentalmente e fornita dai costruttori.

Per dimensionare correttamente i motori, è necessario verificare, che la coppia richiesta in ogni condizione di funzionamento, sia contenuta nella zona di funzionamento del motore, ovvero sia inferiore alla curva di pull-out.

Tramite simulazione è possibile ricavare le coppie T_x e T_y attorno agli assi X e Y del sistema di riferimento R_0. Per conoscere le coppie richieste ai motori è necessario determinare le coppie T_1 T_2 e T_3 riferite ai tre motori, in funzione delle coppie T_x e T_y.

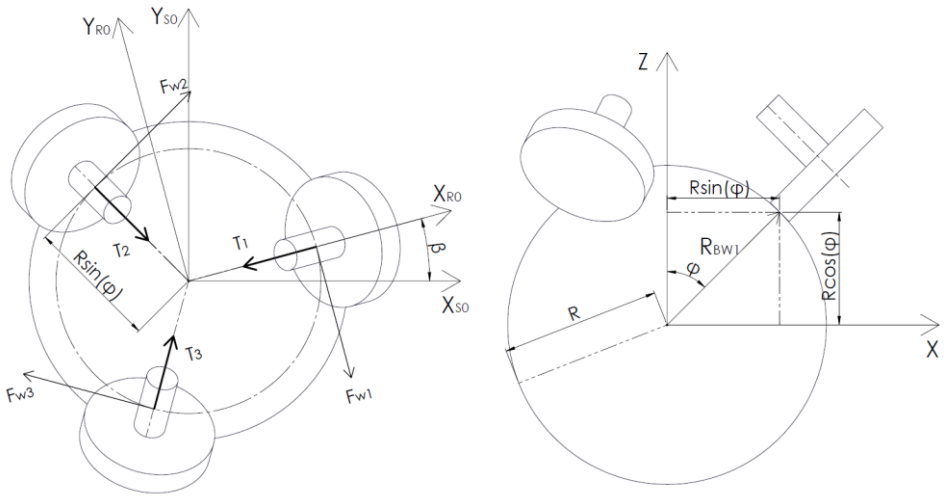


Figura 44: Modello di coppia

In riferimento alla Figura 44 si definiscono alcuni parametri utili per il calcolo della coppia motrice:

F_{Wi}: Forza tangenziale scambiata nel punto i-esimo di contatto ruota-sfera
T_{Bi}: Coppia, in riferimento alla sfera, trasferita alla sfera dalla ruota i-esima.
T_i: Coppia, in riferimento alla ruota, erogata dalla ruota i-esima.
R_{BWi}: Vettore che congiunge il centro sfera con il punto di contatto i-esimo.

È quindi possibile definire:

$$T_{B,1} = R_{B,1} \times F_{W,1}$$

$$T_{B,2} = R_{B,2} \times F_{W,2}$$

$$T_{B,3} = R_{B,3} \times F_{W,3}$$

Inoltre, la forza d'attrito F_W può essere scritta:

$$F_{W,1} = \frac{T_1}{r} \cdot \begin{bmatrix} -\sin\beta \\ \cos\beta \\ 0 \end{bmatrix}$$

$$F_{W,2} = \frac{T_2}{r} \cdot \begin{bmatrix} -\sin\left(\frac{2\pi}{3} - \beta\right) \\ \cos\left(\frac{2\pi}{3} - \beta\right) \\ 0 \end{bmatrix}$$

$$F_{W,3} = \frac{T_3}{r} \cdot \begin{bmatrix} -\sin\left(\frac{2\pi}{3} + \beta\right) \\ \cos\left(\frac{2\pi}{3} + \beta\right) \\ 0 \end{bmatrix}$$

Infine il vettore congiungente il centro ruota e il punto di contatto, risulta:

$$R_{BW,1} = R \cdot \begin{bmatrix} -\sin\beta \\ \cos\beta \\ 0 \end{bmatrix}$$

$$R_{BW,2} = R \cdot \begin{bmatrix} \sin\varphi\cos\left(\frac{2\pi}{3} - \beta\right) \\ \sin\varphi\sin\left(\frac{2\pi}{3} - \beta\right) \\ \cos\varphi \end{bmatrix}$$

$$R_{BW,3} = R \cdot \begin{bmatrix} \sin\varphi\cos\left(\frac{2\pi}{3} + \beta\right) \\ \sin\varphi\sin\left(\frac{2\pi}{3} + \beta\right) \\ \cos\varphi \end{bmatrix}$$

Scrivendo invece la coppia generata dalle ruote equivalenti del modello semplificato, risulta:

$$T_{B,x} = R_{B,x} \times F_{Wx}$$

$$T_{B,y} = R_{B,y} \times F_{Wy}$$

$$T_{B,z} = R_{B,z} \times F_{Wz}$$ [4]

[4] L'ultima equazione ci interessa relativamente poiché abbiamo considerato l'attrito tra il terreno e la sfera sufficiente a impedire la rotazione della sfera attorno all'asse Z.

Inoltre, la forza di attrito nel modello semplificato, risulta:

$$F_{Wx} = \frac{T_x}{r} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$F_{Wy} = \frac{T_y}{r} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$F_{Wz} = \frac{T_z}{r} \cdot \begin{bmatrix} -\sin\beta \\ \cos\beta \\ 0 \end{bmatrix}$$

Infine i raggi nel modello semplificato risultano:

$$R_{BWx} = R_{BWy} = R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$R_{BWz} = R \begin{bmatrix} 0 \\ 0 \\ \cos\varphi \end{bmatrix}$$

Affinchè sia valido il modello, la coppia generata dalle ruote equivalenti deve corrispondere alla coppia generata dalle tre ruote del sistema reale, pertanto risulta:

$$T_{B1} + T_{B2} + T_{B3} = T_{Bx} + T_{By} + T_{Bz}$$

Risolvendo il sistema seguente

$$\frac{R}{r}\left(T_1 \begin{vmatrix} i & j & k \\ sin\varphi cos\beta & sin\varphi sin\beta & cos\varphi \\ -sin\beta & cos\beta & 0 \end{vmatrix} \right.$$

$$+ T_2 \begin{vmatrix} i & j & k \\ sin\varphi\left(-\frac{1}{2}cos\beta + \frac{\sqrt{3}}{2}sin\beta\right) & sin\varphi\left(-\frac{\sqrt{3}}{2}cos\beta - \frac{1}{2}sin\beta\right) & cos\varphi \\ \frac{\sqrt{3}}{2}cos\beta + \frac{1}{2}sin\beta & -\frac{1}{2}cos\beta + \frac{\sqrt{3}}{2}sin\beta & 0 \end{vmatrix}$$

$$\left. + T_3 \begin{vmatrix} i & j & k \\ sin\varphi\left(-\frac{1}{2}cos\beta - \frac{\sqrt{3}}{2}sin\beta\right) & sin\varphi\left(\frac{\sqrt{3}}{2}cos\beta - \frac{1}{2}sin\beta\right) & cos\varphi \\ -\frac{\sqrt{3}}{2}cos\beta + \frac{1}{2}sin\beta & -\frac{1}{2}cos\beta - \frac{\sqrt{3}}{2}sin\beta & 0 \end{vmatrix} \right) =$$

$$= \frac{R}{r}\left(T_x \begin{vmatrix} i & j & k \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{vmatrix} + T_y \begin{vmatrix} i & j & k \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{vmatrix} + T_z \begin{vmatrix} i & j & k \\ cos\beta sin\varphi & sin\beta cos\varphi & 0 \\ -sin\beta & cos\beta & 0 \end{vmatrix} \right)$$

Risulta:

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} cos\varphi cos\beta & -cos\varphi\left(\frac{1}{2}cos\beta - \frac{\sqrt{3}}{2}sin\beta\right) & -cos\varphi\left(\frac{1}{2}cos\beta + \frac{\sqrt{3}}{2}sin\beta\right) \\ -cos\varphi sin\beta & cos\varphi\left(\frac{1}{2}sin\beta + \frac{\sqrt{3}}{2}cos\beta\right) & cos\varphi\left(\frac{1}{2}sin\beta - \frac{\sqrt{3}}{2}cos\beta\right) \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

Invertendo la matrice e pre-moltiplicando entrambi i membri per l'inversa è possibile ricavare la coppia motrice delle ruote in funzione della coppia sulla sfera nei tre piani del modello 2D semplificato.

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} \frac{2cos\beta}{cos\varphi} & -\frac{2sin\beta}{cos\varphi} & 1 \\ \frac{1}{cos\varphi}\left(\sqrt{3}sin\beta - cos\beta\right) & \frac{1}{cos\varphi}\left(sin\beta + \sqrt{3}cos\beta\right) & 1 \\ \frac{1}{cos\varphi}\left(-\sqrt{3}sin\beta - cos\beta\right) & \frac{1}{cos\varphi}\left(sin\beta - \sqrt{3}cos\beta\right) & 1 \end{bmatrix} \times \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

Per diverse simulazioni di traslazione e di disturbo, avendo impostato la modalità half-step, è stata ricavata la massima coppia richiesta ai motori e la massima velocità in termini di impulsi al secondo.

X set [m]	**Y set [m]**	**Coppia massima [Nm]**	**velocità massima [pps]**	**Disturbo [N]**
1	0	0.25	600	0
0	1	0.35	680	0
1	1	0.45	930	0
0	2	0.5	1200	0
0	0	0.7	245	30 dir. x
0	0	0.9	225	30 dir. y

Paragonando tale dato con la curva di pull-out del motore rappresentata in Figura 45, si noti come il motore sia in grado di fornire la coppia richiesta dal sistema per le prove in esame. Questo è un buon risultato poiché lascia un discreto margine per sopperire agli errori introdotti nel modello non avendo considerato alcun tipo di attrito. Inoltre, modificando i guadagni del controllo PD relativo alla posizione, è possibile abbassare ulteriormente i punti di funzionamento a discapito della risposta del sistema a un input.

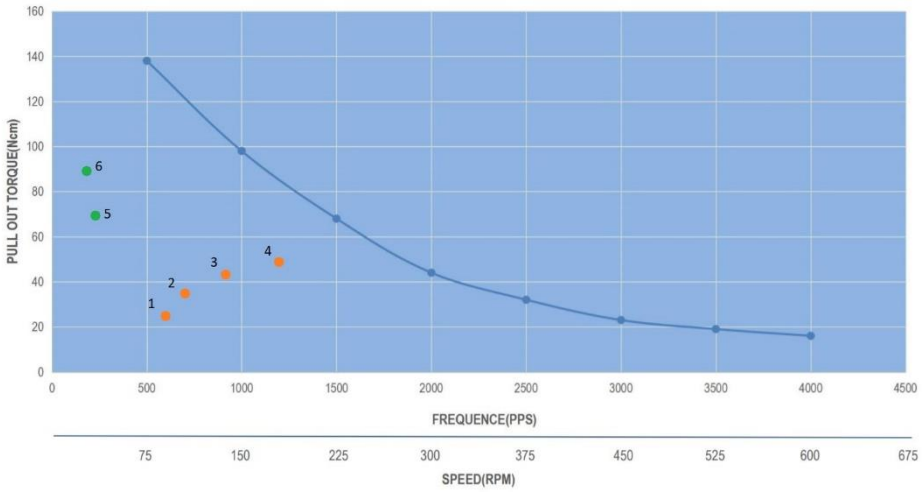


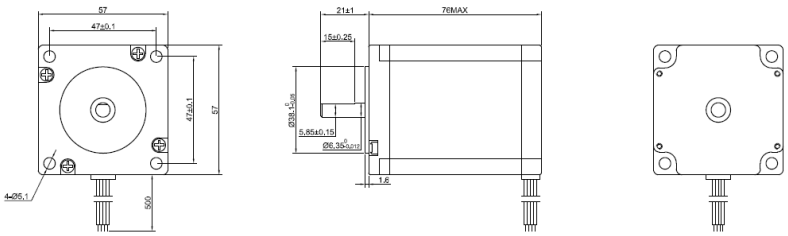
Figura 45: Curva di pull-out

Il motore scelto risulta idoneo ai parametri prestazionali minimi richiesti[5].

Dati caratteristici del motore:

tipo	**avvolg.**	**tens. nom. [V]**	**corrente nominale [A]**	**coppia di ritenuta [Nm]**	**fili esterni**	**Flangia**
Stepper ibrido	bipolare	3.2	2.8	1.9	4 fili	Nema 23

[5] L'analisi effettuata serve a dimostrare che è possibile effettuare simulazioni di traslazione del robot entro un certo raggio con i motori scelti. Si tenga in considerazione però, che la coppia richiesta ai motori è proporzionale all'accelerazione imposta alla sfera, calcolata dal controllo. Variando pertanto i guadagni del controllo, è possibile riscontrare coppie richieste maggiori di quelle rilevate in queste simulazioni. Pertanto l'analisi quantitativa delle coppie perde di significato, resta valida l'analisi qualitativa dalla quale emerge che, regolando opportunamente il controllo le coppie richieste rientrano nella specifica richiesta.



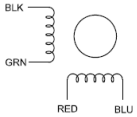
SPECIFICATION / CONNECTION	BIPOLAR
AMPS/PHASE	2.80
RESISTANCE/PHASE(Ohms)@25°C	1.13±10%
INDUCTANCE/PHASE(mH)@1KHz	5.40±20%
HOLDING TORQUE(Nm)[lb-In]	1.89[16.73]
STEP ANGLE(°)	1.80
STEP ACCURACY(NON-ACCUM)	±5.00%
ROTOR INERTIA(g-cm²)	440.00
WEIGHT(Kg)[lb]	1.10[2.43]
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL;FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm（UNDER NORMAL TEMPERATURE AND HUMIDITY）	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

TYPE OF CONNECTION (EXTERN)		MOTOR	
PIN NO	BIPOLAR	LEADS	WINDING
1	A —	BLK	A
2	A\ —	GRN	A\
3	B —	RED	B
4	B\ —	BLU	B\

FULL STEP 2 PHASE-Ex. ,
WHEN FACING MOUNTING END (X)

STEP	A	B	A\	B\
1	+	+	-	-
2	-	+	+	-
3	-	-	+	+
4	+	-	-	+

CCW ↑ / ↓ CW

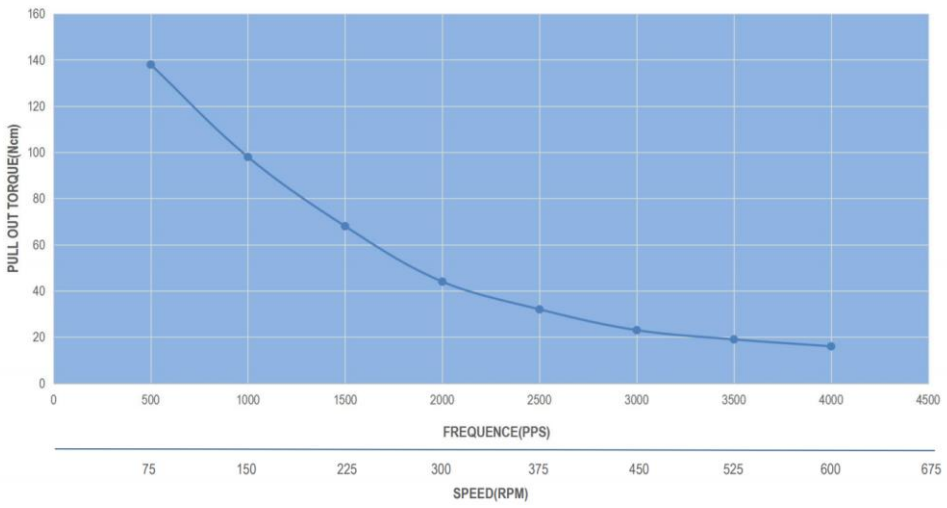


STEPPERONLINE®		APVD		8.18.2018
		CHKD		
	1:1.5	DRN		
	SCALE	SIGNATURE		DATE

STEPPER MOTOR

23HS30-2804S

23HS30-2804S PULL OUT TORQUE(2.8A/P 24V HALF STEP)



6.2 Driver

La curva di pull-out fornita dal costruttore per il motore scelto, si riferisce unicamente al caso di tensione di alimentazione pari a 24 V, corrente di armatura pari a 2.8 A e azionamento a mezzo passo. Tali parametri influenzano fortemente la curva, poiché la coppia erogabile dal motore è proporzionale all'intensità di corrente negli avvolgimenti, che a sua volta è proporzionale alla tensione di alimentazione. Per tensioni di alimentazione troppo basse il transitorio di corrente non è più trascurabile rispetto la frequenza degli impulsi e la corrente non riuscirebbe a raggiungere il valore nominale.

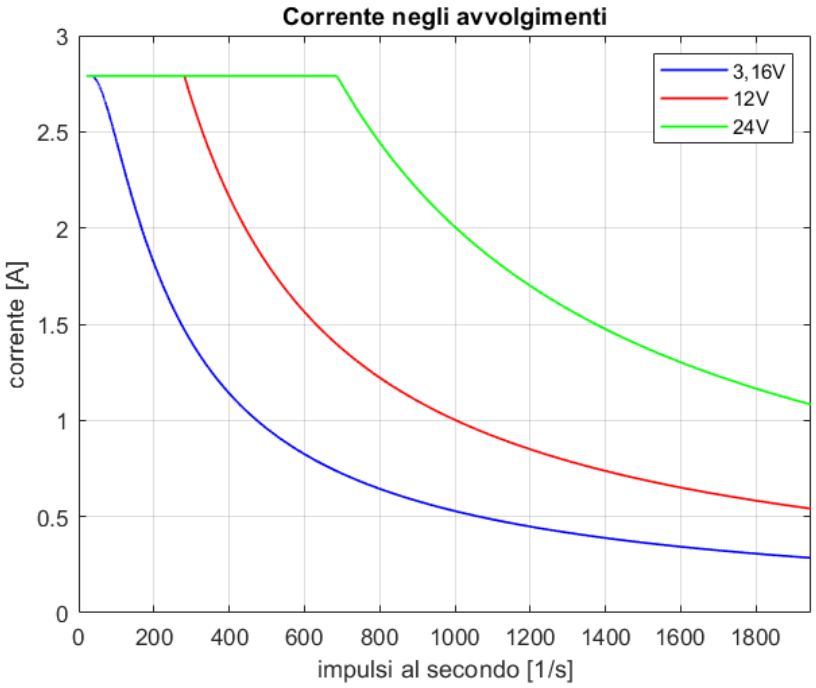


Figura 46: Andamento della corrente in funzione della frequenza di alimentazione

In Figura 46 è mostrato l'andamento della corrente avendo simulato la chiusura di un circuito RL con i valori di resistenza e induttanza pari ai valori presenti sui dati di targa del motore scelto. Per semplicità si è scelto un valore un valore massimo della corrente pari a 2.8 A che

corrisponde alla corrente nominale negli avvolgimenti quando il motore è guidato dal driver a passo intero in modalità one phase on[6].

Si noti come, per valori di tensione decrescenti, il valore di corrente raggiunta diminuisca drasticamente all'aumentare della frequenza di alimentazione. Il valore di corrente nell'avvolgimento, alimentando a 24 V, è ricavabile dalla legge di ohm :

$$I = \frac{V}{R} = \frac{24}{1.13} = 21.24\ A$$

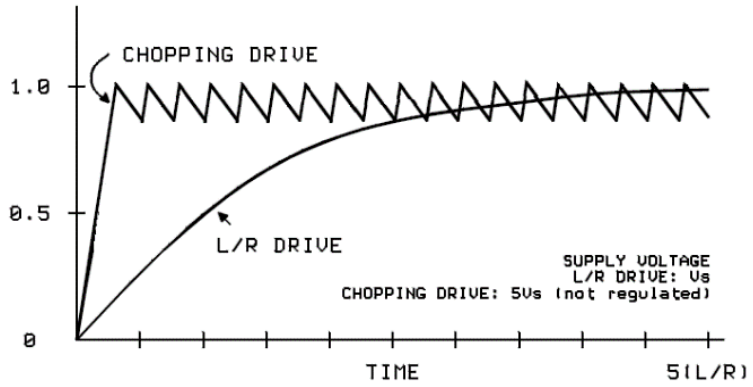


Figura 47: Andamento della corrente con alimentazione PWM

Questo valore di tensione è nettamente superiore a quello riportato nei dati di targa del motore, pari a 2.8 A; il motore, sottoposto a tale corrente, si danneggerebbe irrimediabilmente. Pertanto, è necessario limitare tale valore di corrente al valore nominale. A tal fine, si sceglie un driver che presenti il controllo di corrente.

In Figura 47, è mostrato un esempio di andamento della corrente nella fase, nel caso di alimentazione con tensione nominale e nel caso di tensione di alimentazione maggiore. Nel secondo caso, quando il valore di corrente raggiunge il valore impostato, il driver alimenta la fase in modalità PWM con una frequenza portante fissa, mantenendo costante la corrente. In questo modo è possibile alimentare il motore con tensioni molto maggiori della tensione

[6] Nella pratica, molti driver in commercio, utilizzano una modalità di comando two phase on che consiste nell'alimentare contemporaneamente 2 fasi a un valore di corrente pari a 70.71% (corrispondente a $\frac{1}{\sqrt{2}}$) del valore nominale. Questo consente di ottenere un valore di coppia maggiore di un fattore $\sqrt{2}$ rispetto quello ottenuto alimentando one phase on.

nominale. Ne consegue che il limite di tensione di alimentazione è imposto dai dati di targa del driver.

Pertanto, i dati di targa minimi del driver sono:

- Tensione massima ≥ 24V
- Corrente elaborabile ≥ 2.8A
- Controllo di corrente integrato

È importante che il valore di corrente elaborabile sia sufficientemente maggiore del valore nominale per evitare surriscaldamenti del driver.

È stato individuato un driver che soddisfi tutti i parametri minimi richiesti:

Driver TOSHIBA TB6600

Direzione	Possibilità di controllare CW e CCW
Tensione massima [V]	50
Corrente massima [A]	4.5
Modalità di azionamento disponibile	$\frac{1}{32}\ \frac{1}{16}\ \frac{1}{8}\ \frac{1}{4}\ \frac{1}{2}\ \ 1\ passo$

6.3 Microcontrollore

Per quanto riguarda la parte hardware di elettronica, si è scelto di utilizzare due schede Arduino operanti in parallelo. Arduino è una piattaforma elettronica open-source che mette a disposizioni componenti hardware intuitivi e facili da utilizzare. Il grande vantaggio nell'utilizzare questo tipo di microcontrollore è la praticità di utilizzo e di programmazione delle schede grazie alle numerose librerie messe gratuitamente a disposizione degli utenti.

Figura 48: Arduino UNO Rev3

La scheda utilizzata è Arduino UNO Rev3 mostrata in Figura 48, si è scelto di utilizzare questa scheda, almeno per le prime fasi di messa in moto del progetto per il suo basso costo e la sua semplicità di messa in funzione. Questa scheda è basata su processore ATmega328P con una frequenza di clock di 16MHz, Grazie alla standardizzazione adottata dalla azienda Arduino e alla compatibilità delle schede sarà possibile, in una fase successiva, sostituire la scheda con una più performante.

In parallelo alla scheda Arduino UNO Rev3 è stata utilizzata un'altra scheda Arduino, la Arduino MEGA2560P. A differenza della prima, questa scheda presenta molti più pin disponibili e una memoria più ampia. La disponibilità di numerosi pin potrà essere utile per sviluppi futuri del robot dando la possibilità di aggiungere numerosi accessori e controllarli con la scheda.

6.4 Modulo di lettura inclinazione

Si utilizza il modulo GY-521 il quale gestisce il chip MPU6050 prodotto dalla InvenSense™, il quale combina al suo internoi:

Accelerometro a 3 assi: La funzione dell'accelerometro è quella di leggere l'angolo di inclinazione lungo gli assi principali del dispositivo mediante un sistema elettro meccanico (MEMS). Inoltre rileva l'accelerazione lungo gli assi normalizzando la misura all'accelerazione gravitazionale, questo significa che posizionato su un piano orizzontale i valori misurati sono [0, 0, 1]. I valori misurati dal MEMS vengono digitalizzati dal DAC a 16bit con la possibilità di programmare gli intervalli di uscita a fondo scala con i seguenti valori: ±2g; ±4g; ±8g; ±16g

Giroscopio a 3 assi: Rileva la velocità di rotazione attorno alla terna del sistema di riferimento. La digitalizzazione avviene sempre a 16bit con la possibilità di selezionare gli intervalli di uscita a fondo scala: ±250; ±500; ±1000; ±2000. L'unità di misura in output è $\left[\frac{°}{s}\right]$.

Digital motion processor: Il processore di movimento digitale™ sviluppato dalla InvenSense viene utilizzato per eseguire calcoli sui dati del sensore. Il DMP™ è integrato nel chip MPU-6050 il quale include anche un buffer FIFO da 1024 byte dove vengono immagazzinati i dati. L'MPU-6050 funziona come Slave quando è connesso all'arduino con i pin SDA e SCL collegati al bus I^2C.

Sono inoltre presenti un **oscillatore di clock interno** che rende flessibile lo schema di temporizzazione e un **sensore di temperatura**.

6.5 Omniwheel

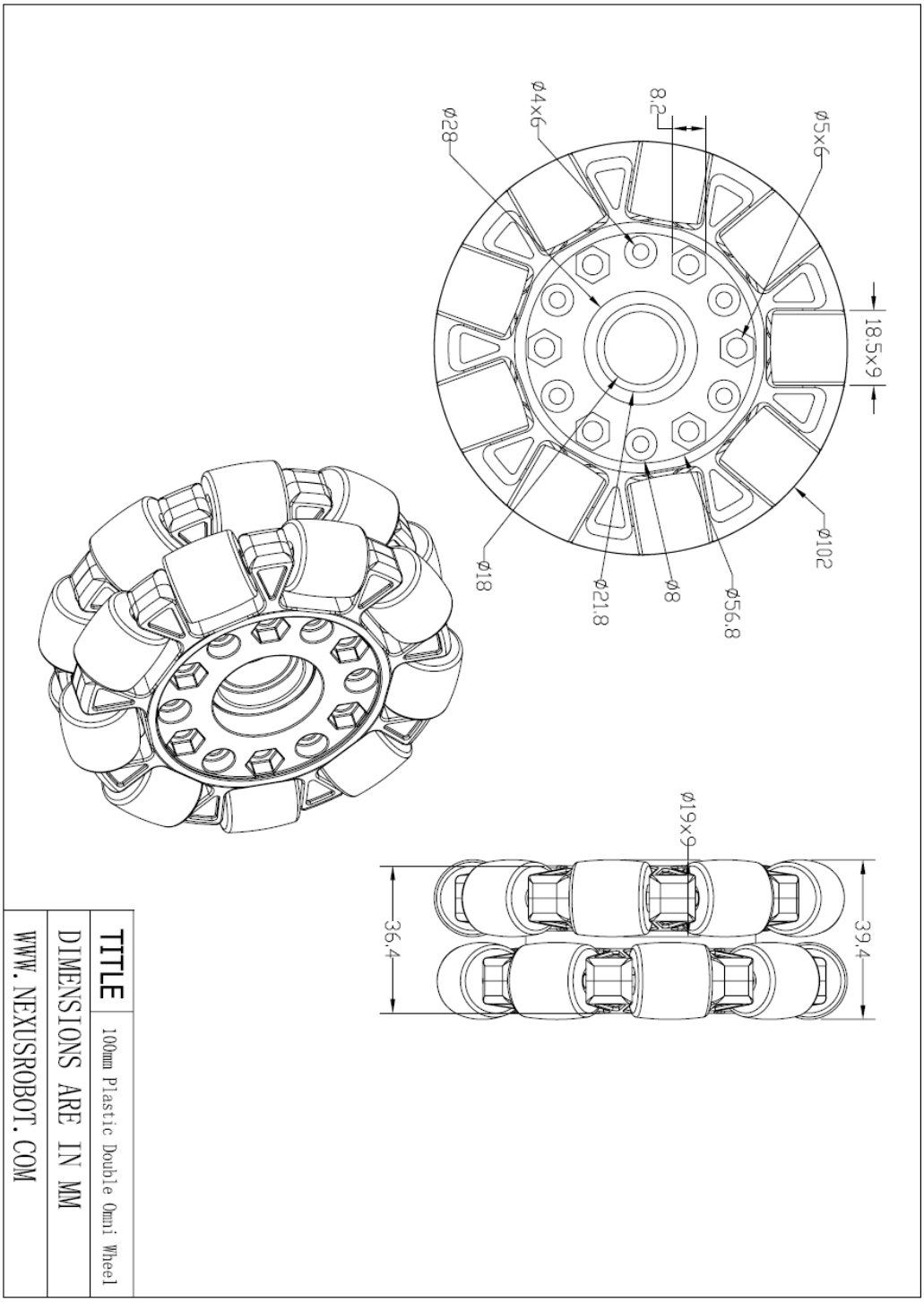


Figura 49: 100mm omnidirectional wheel

Figura 50: Ruota omnidirezionale utilizzata

Si è scelto di utilizzare una ruota omnidirezionale a due file di rulli, questa ruota soffre dell'errore introdotto dall'algoritmo di cinematica approssimata adottato nella parte software, tuttavia, considerando la cedevolezza della struttura del robot e le ampie tolleranze di montaggio, il punto di appoggio avviene contemporaneamente su entrambe le file di rulli. Pertanto si è deciso di non appesantire la parte software con un algoritmo di cinematica accurato. Le prestazioni di questa ruota sono accettabili per una prima versione di ballbot.

6.6 Controller

Per il comando del robot si è scelto di utilizzare un comune controller da console la cui lettura viene effettuata dalla scheda Arduino MEGA 2560P mediante l'utilizzo della libreria Psx.h.

I comandi sono stati impostati in modo tale da poter fornire in input sia dei comandi a gradino di ampiezza proporzionale al tempo di pressione del tasto, sia dei segnali a rampa, ovvero un set di posizione in input che si muove nello spazio a velocità costante. Sono stati settati due ulteriori tasti per permettere la rotazione oraria e antioraria del robot sulla sfera.

7 MODELLI CAD

In questo capitolo vengono riportati i modelli CAD dimensionati e disegnati in funzione delle specifiche richieste.

Il modello completo delle parti essenziali al funzionamento è rappresentato in Figura 51.

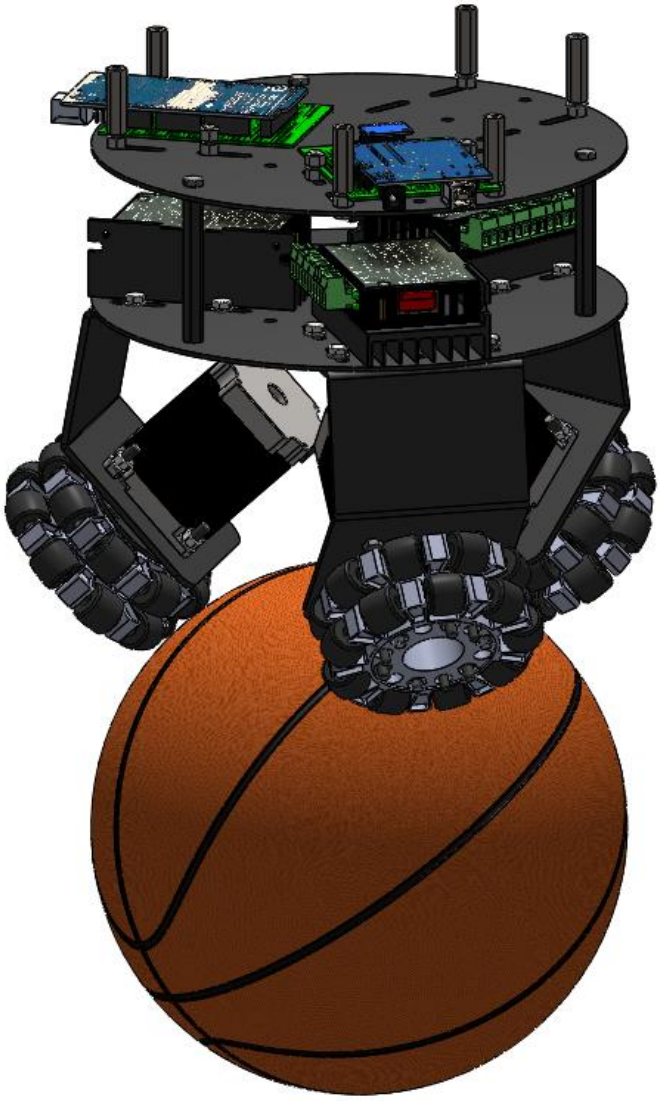


Figura 51: Modello CAD del ballbott

7.1 Piastra superiore

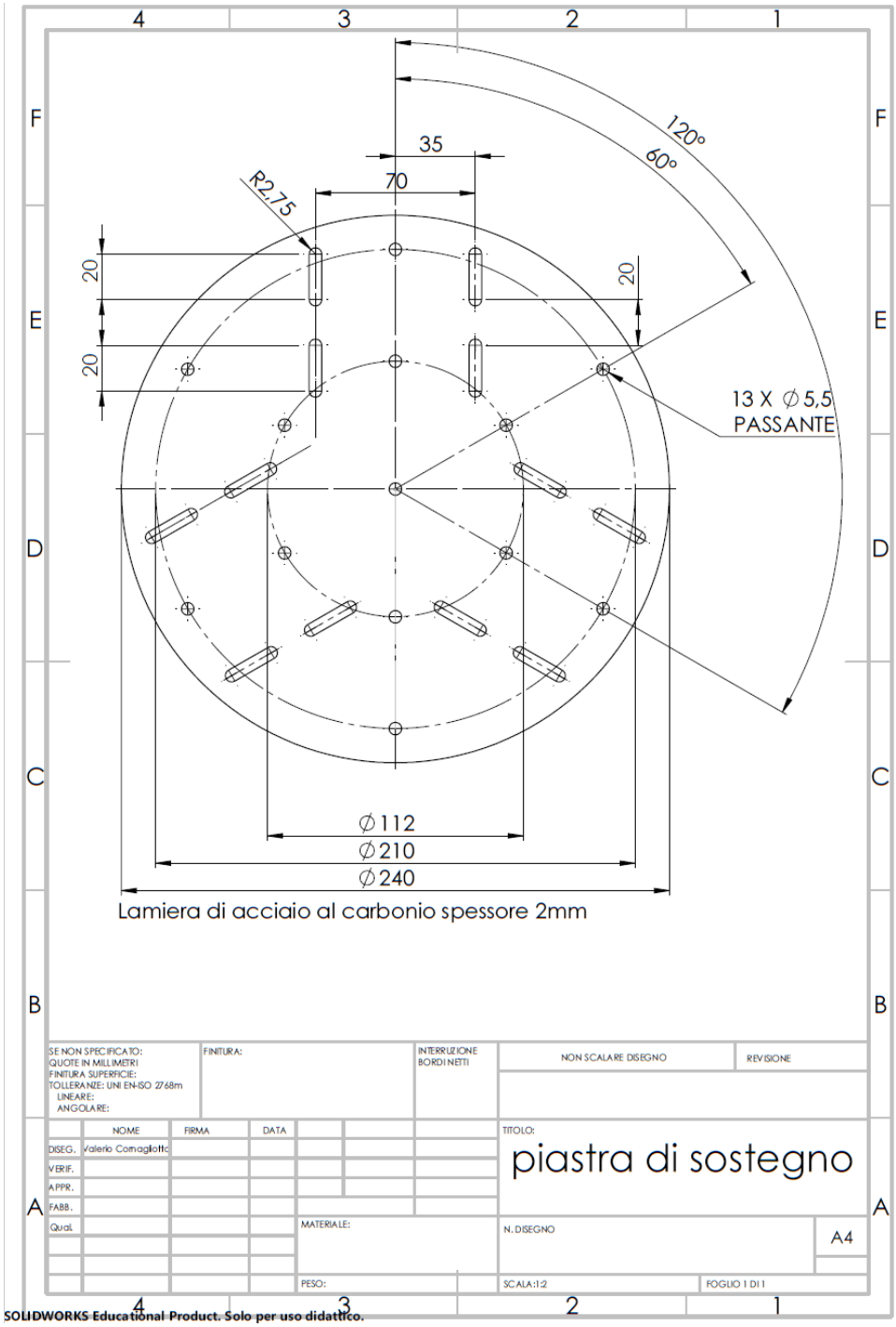


Figura 52: Piastra di sostengo

Sulla piastra sono ricavate le asole per vite passante M5 utili per il fissaggio dei supporti motore con angolazione di 120° l'uno dall'altro. Sono state praticate le asole invece che semplici fori per permettere di regolare i supporti in modo tale che le ruote siano perfettamente tangenti alla sfera anche nel caso di utilizzo di sfere di diametri diversi da quello di progetto.

Sono stati inoltre ricavati dei fori per vite passante M5 per rendere modulabile le piastre del robot. In questo modo si possono aggiungere elementi semplicemente collegando le piastre una sull'altra. Ulteriori fori servono per eventuali ganci di supporto almeno per le prime prove di tuning dei guadagni.

La scelta del diametro è stata fatto considerando l'utilizzo di un pallone da pallacanestro di diametro pari a 240mm. Si è scelto di utilizzare una sfera di questo tipo per il materiale gommoso della superficie che garantisce un buon coefficiente di attrito con i rulli delle ruote omnidirezionali.

7.2 Supporto motore

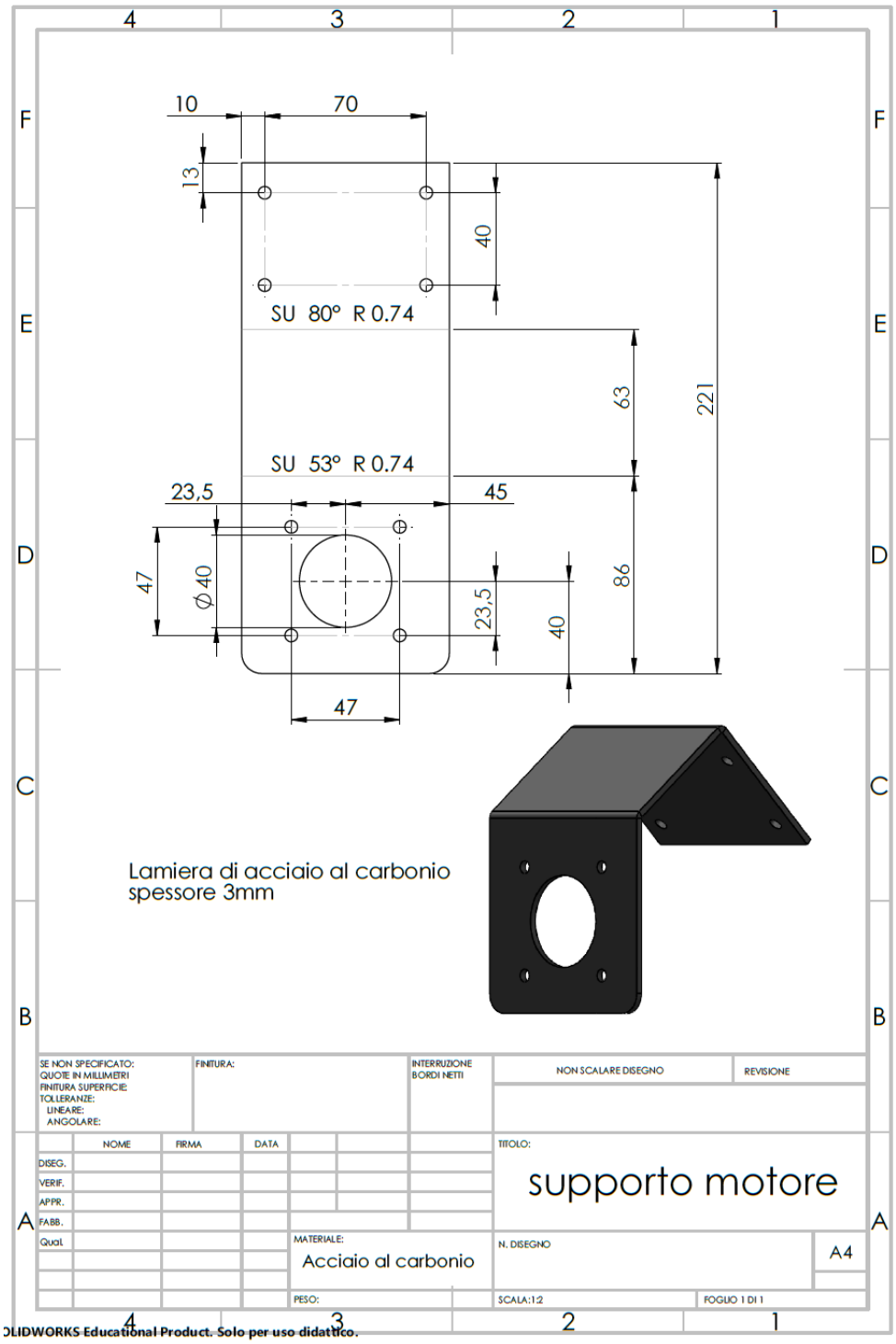


Figura 53: supporto motore

Considerando le ruote omnidirezionali di diametro pari a 102mm e la flangia standardizzata del motore NEMA 23, si è dimensionato il supporto motore e l'angolo di contatto delle ruote con la sfera.

Un parametro fondamentale per il dimensionamento del supporto e del mozzo di collegamento tra la ruota e il motore, è il carico radiale massimo applicabile all'albero motore.

I dati di targa del motore riportano un valore di 75N a una distanza di 20 mm dalla flangia, corrispondente a un momento di 1,5 Nm. È fondamentale rispettare tale vincolo poiché è il carico massimo dei cuscinetti che sorreggono il rotore.

Rappresentando l'andamento della forza radiale massima in funzione della distanza di applicazione della stessa, è possibile verificare che il dimensionamento effettuato sia congruente con le specifiche.

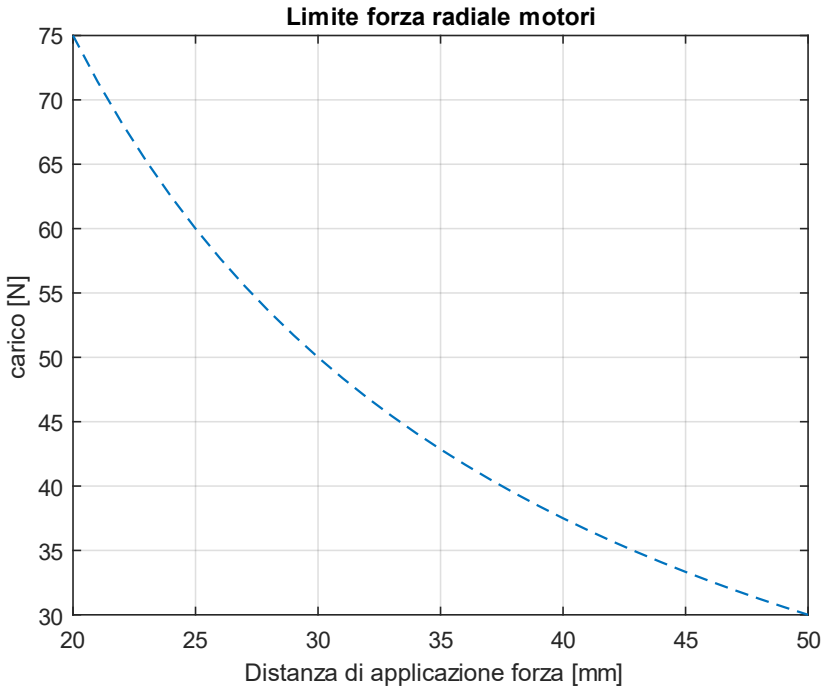


Figura 54: Carico limite motori

Dato un peso del robot di 8kg, calcolato secondo la configurazione prototipale attuale, è possibile confrontare il carico sui motori con il carico limite, al variare dell'angolo di inclinazione φ delle ruote.

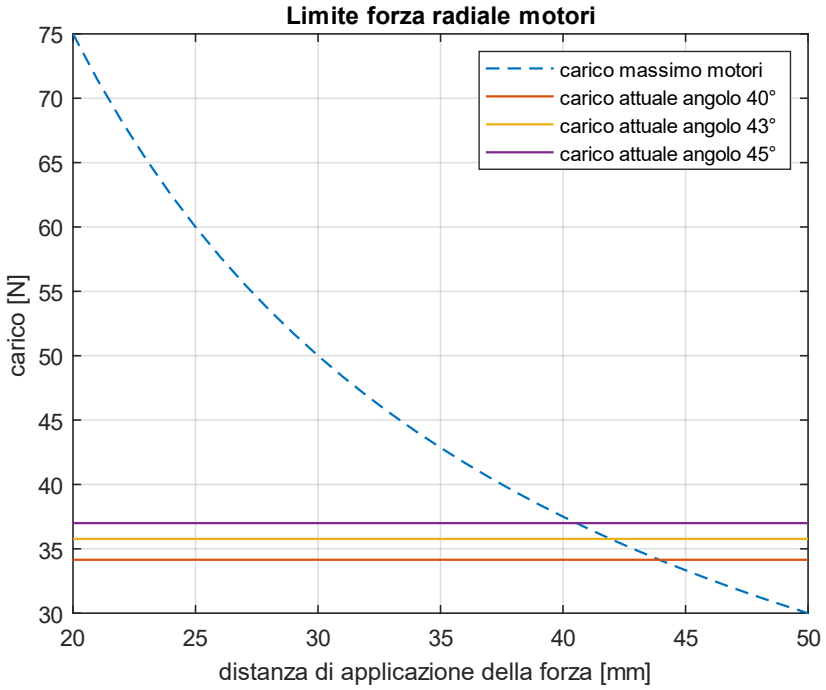


Figura 55: Carico limite e carico attuale

Come si può notare dalla Figura 55, il carico che grava sull'asse motore in direzione radiale, calcolato a parità di peso (considerando un totale di 8 Kg), aumenta all'aumentare dell'angolo di inclinazione delle ruote. Sarebbe pertanto opportuno diminuire il più possibile tale angolo. Tuttavia i vincoli geometrici impongono un angolo limite oltre il quale si creerebbe interferenza geometrica nella parte posteriore dei motori come visibile in Figura 56. Si è scelto dunque, un angolo di inclinazione pari a 43°.

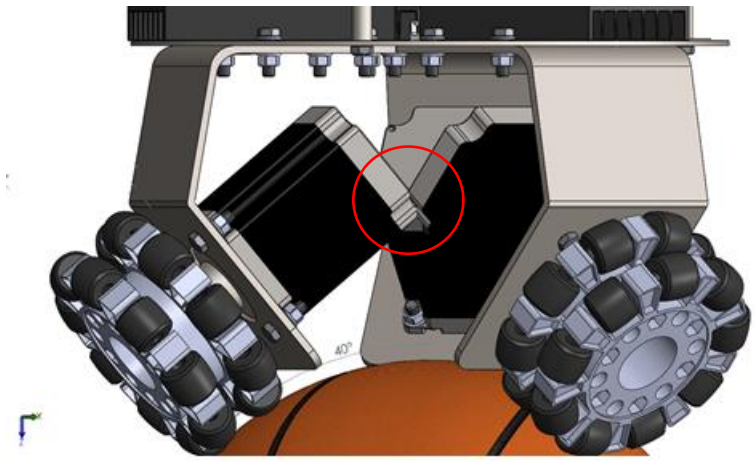


Figura 56: Interferenza motori per inclinazione ruote di 40°

Al fine di rientrare quanto più possibile nelle specifiche di carico, si è dimensionato il mozzo di collegamento motore-ruota in modo tale da ridurre al minimo la distanza di applicazione del carico.

Per verificare che lo spessore e la forma dei supporti fossero idonei al peso del robot, è stata effettuata una simulazione mediante il tool SimulationXpress presente su Solidworks™. È risultato sufficiente uno spessore della lamiera di 3mm per limitare la freccia massima a un valore inferiore a 1mm; accettabile per il contatto ruota sfera.

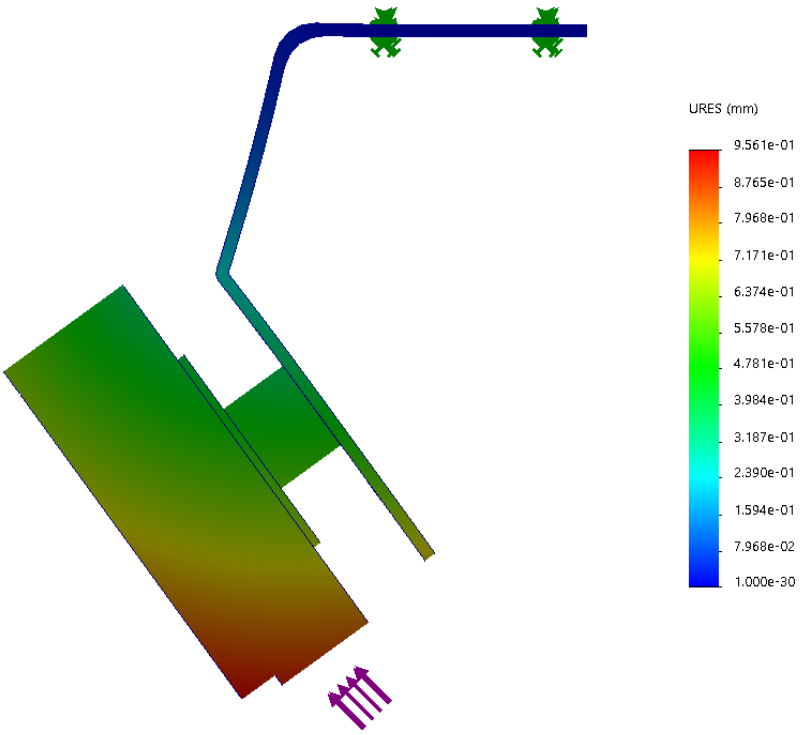


Figura 57: Simulazione deformazione supporto

Non potendo effettuare analisi multibody con il software è stata creata una parte che simulasse la presenza della ruota ed è stato applicato il carico come se la ruota poggiasse sulla fila di rulli esterna. Nei fori superiori del supporto è stato messo un vincolo ad incastro. In Figura 57 è visibile la simulazione effettuata per uno spessore di lamiera pari a 3mm.

7.3 Mozzo con flangia di collegamento

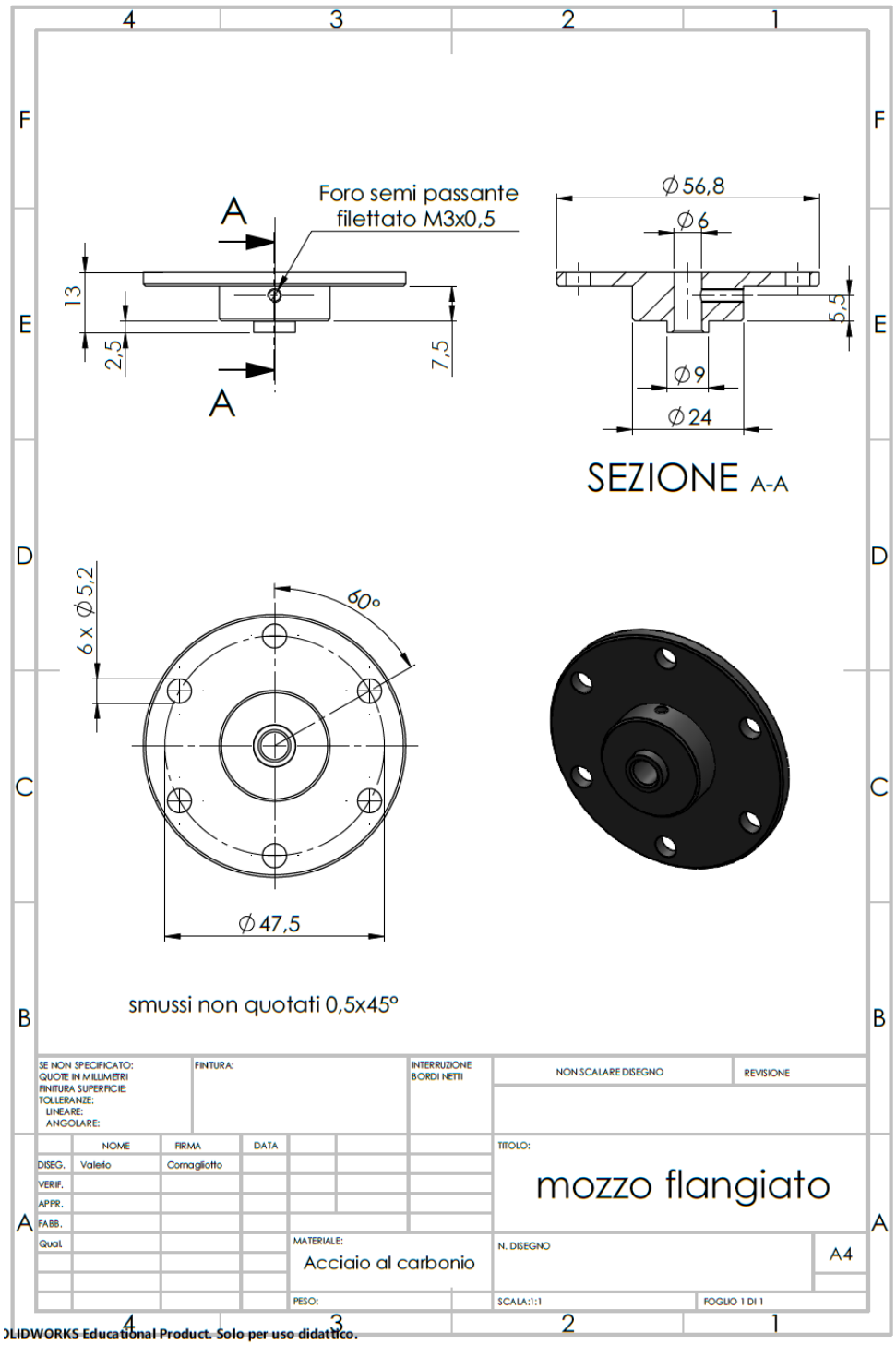


Figura 58: Mozzo di collegamento albero-ruota

Per valutare la distanza di applicazione del carico è necessario fare riferimento alla Figura 59, in cui vengono rappresentate le misure fondamentali[7] utili per l'analisi. Viene misurata la distanza dal piano mediano dei rulli alla flangia del motore, per entrambe le file di rulli.

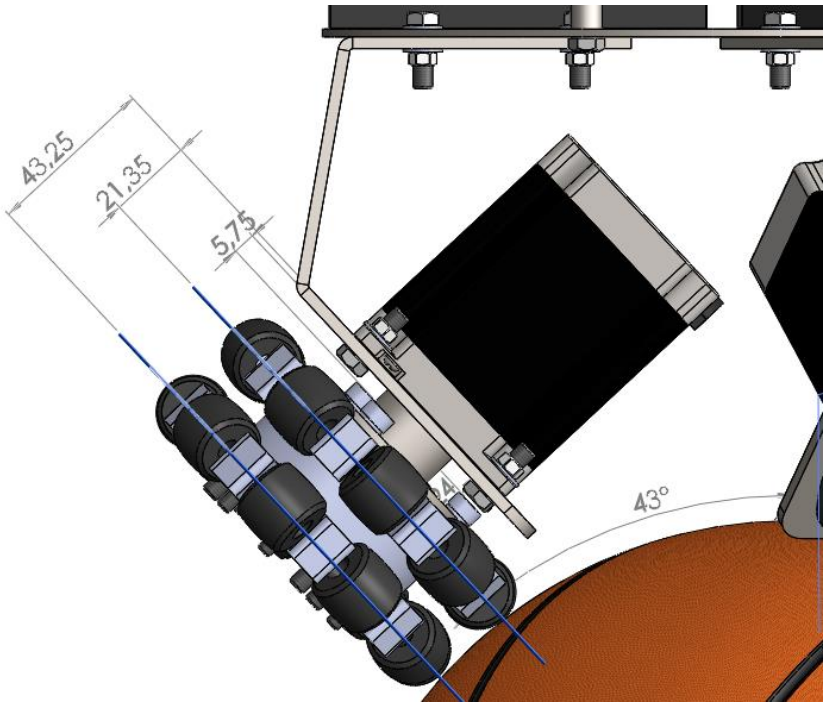


Figura 59: Distanza di applicazione del carico

Le ruote omnidirezionali, sono progettate in maniera tale che, per ogni istante, vi sia un solo rullo a contatto con la sfera. Durante il rotolamento il contatto si sposta dalla fila interna alla fila esterna e viceversa. Pertanto la distanza di applicazione del carico varia notevolmente da un caso all'altro.

Rulli interni: 21mm

Rulli esterni: 43 mm

[7] La misura 5,75 mm si riferisce alla distanza che c'è tra il foro filettato del mozzo per il collegamento con l'albero e la flangia del motore, questa distanza deve risultare almeno 5 mm in quanto è la distanza a cui inizia la spianatura sull'albero motore.

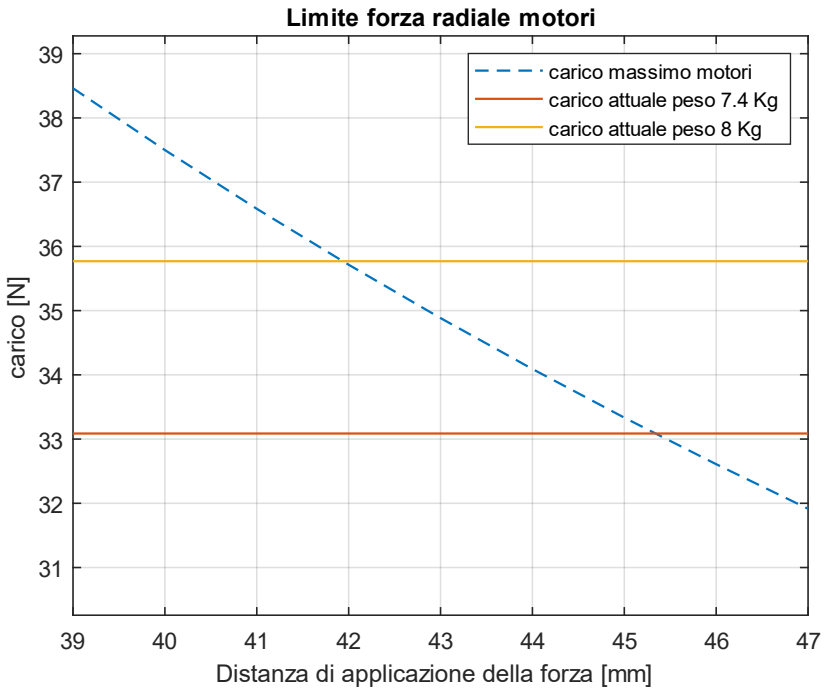


Figura 60: Limiti di carico

Facendo riferimento alla Figura 55 si noti che per quanto riguarda la distanza di applicazione relativa al rullo interno ci troviamo entro i limiti imposti dal motore con ampio margine; risulta invece critico il contatto in prossimità della fila esterna, il quale avviene a una distanza di circa 43mm.

Oltre aver dimensionato il mozzo in maniera tale da diminuire il più possibile la distanza di applicazione del carico radiale, una modifica ulteriore per poter rientrare nelle specifiche è quella di diminuire il peso totale del robot. Utilizzando materiali più leggeri per il telaio, o eliminando la piastra di supporto superiore è possibile abbassare il peso totale. In Figura 60 è mostrata la differenza di carico qualora si eliminasse la piastra superiore, riducendo il peso da 8 Kg a 7,4 Kg. Nel secondo caso, ad una distanza di 43mm corrispondente alla fila di rulli esterna, il carico radiale sui motori rientra nei limiti imposti dal costruttore dei motori.

Questa analisi è valida nel caso ideale ovvero considerando il sistema infinitamente rigido, e tutte le misure nominali. Tuttavia, essendo la sfera di gomma e considerando una minima

cedevolezza di deformabilità del sistema, è ragionevole pensare che il contatto tra la sfera e la ruota avvenga contemporaneamente sulle due file di rulli. Sotto queste ipotesi il punto di contatto si può considerare come punto medio tra le due file di rulli ovvero a una distanza di 32 mm, rimanendo ampiamente entro i limiti imposti.

Al fine di mantenere un certo margine di sicurezza sul carico, per il dimensionamento dei componenti, si considera il caso ideale, ovvero considerando il punto di contatto singolo sulla fila di rulli esterni che corrisponde alla condizione più critica che si possa manifestare. Cosi facendo sono garantite le corrette condizioni di funzionamento del motore.

7.4 Distanziale

Per collegare le due piastre, si sono utilizzati tre distanziali esagonali con due fori assiali filettati M5 alle estremità.

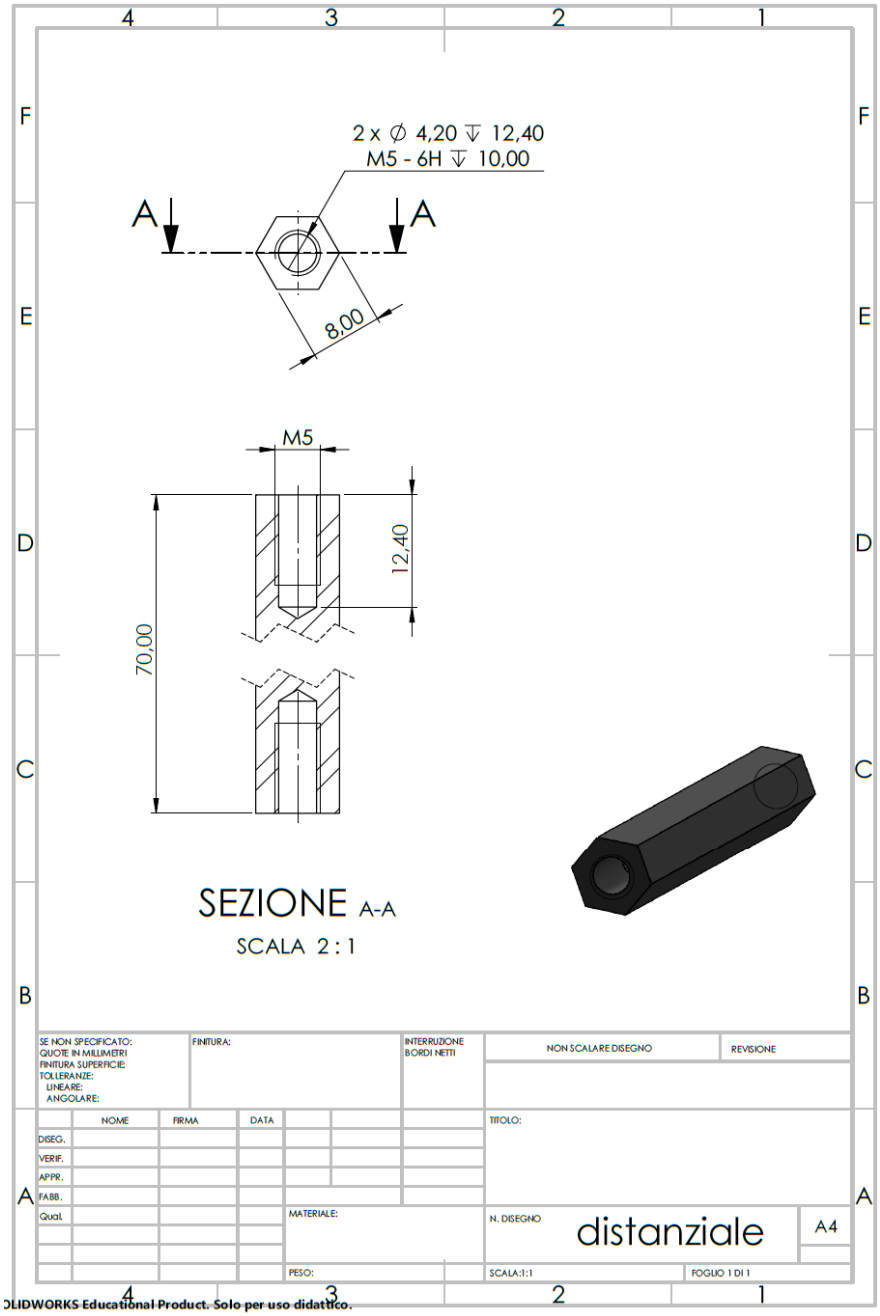


Figura 61: Distanziale

8 SOFTWARE DI AZIONAMENTO E CONTROLLO

La logica di controllo del ballbot viene affidata a due board della famiglia Arduino, nello specifico la scheda Arduino UNO R3 e una Arduino MEGA2560 le quali montano rispettivamente il processore ATmega328P e ATmega2560 entrambi operanti a una frequenza di clock di 16MHz.

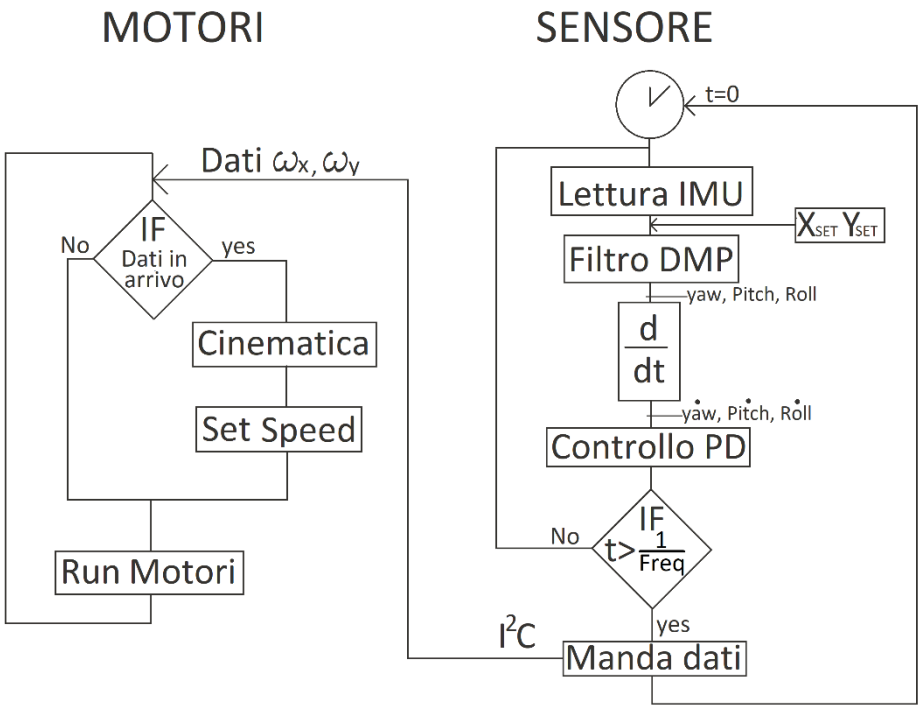


Figura 62: Logica di controllo

In Figura 62 sono rappresentati i due loop di codice effettuati in parallelo dalle due schede. Nella parte destra dell'immagine è rappresentata la logica riguardante il processore preposto alla lettura e all'elaborazione dei dati provenienti dall'accelerometro, mediante il metodo di lettura dei dati noto come DMP (Digital Motion Processor™) sviluppato dalla InvenSense. È stato possibile utilizzare tale metodo grazie al codice fornito da Jeff Rowberg. A differenza del metodo tradizionale di lettura dati, il quale invia i dati in maniera sincrona rispetto la lettura, questo metodo è asincrono, ovvero i dati vengono letti, elaborati dal DMP e immagazzinati nel buffer FIFO della scheda, vengono inviati al Master quando richiesto.

Dopo aver ricavato gli angoli di inclinazione del robot, si calcolano le velocità di inclinazione mediante rapporto incrementale. Tale valore è moltiplicato per il guadagno derivativo all'interno del controllo PD. Mediante la funzione millis() è possibile leggere il tempo trascorso tra due cicli successivi, questo permette di calcolare approssimativamente la posizione del robot in open loop moltiplicando la velocità per il dt trascorso.

In fondo al loop relativo alla scheda per la lettura dei dati è presente un blocco IF. il quale confronta il tempo trascorso con il reciproco della frequenza scelta per inviare i segnali al processore preposto al controllo dei motori. In questo caso si è imposta una frequenza di 100 Hz che può essere cambiata in fase di messa a punto del prototipo modificandone il valore ricompilando la scheda. Tuttavia si è riscontrato che i processori utilizzati a 16Mhz limitano tale valore.

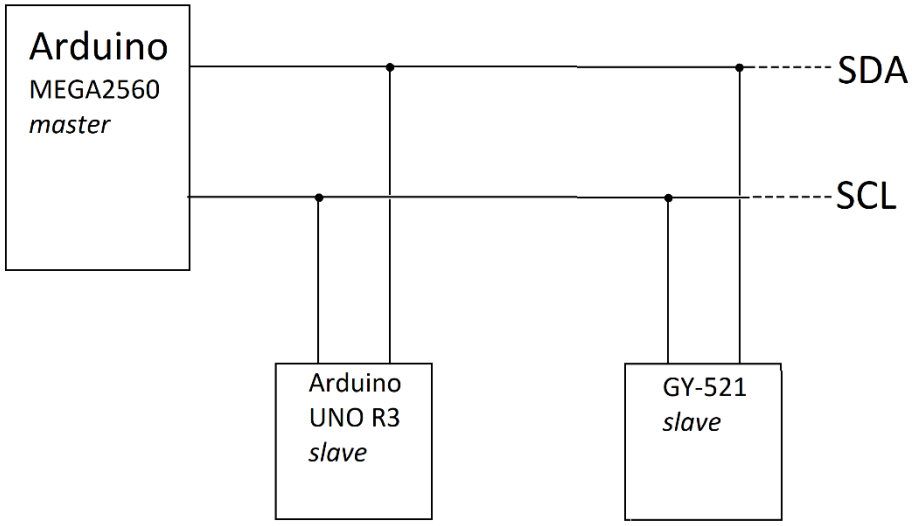


Figura 63: comunicazione ad architettura Master-Slave

I dati vengono inviati sfruttando la suite di protocolli I^2C presente sul processore ATmega. Secondo questo protocollo vi è una scheda "master" e una scheda "slave", in comunicazione mediante due linee comuni visibili in Figura 63. SDA (serial data line) è la linea attraverso cui passano i dati, e SCL (serial clock line) gestisce la frequenza di comunicazione. Ogni qual volta vi siano dei dati in arrivo sulla scheda slave inviati dalla scheda master, il loop principale della scheda slave si interrompe ed esegue una funzione definita nel codice attraverso la

funzione Wire.onReceive( ); al termine dell'esecuzione della funzione, il processore dello slave ricomincia a eseguire il loop principale. La stessa architettura Master-Slave è impiegata dalla scheda GY-521 per inviare i dati alla scheda Master. In questo caso, essendo la scheda Slave il mittente, i pacchetti di dati vengono instradati sui bus di comunicazione solo quando il Master lo richiede. In questo modo è il Master a definire la frequenza di clock di comunicazione con le schede e gestisce il traffico di pacchetti di informazioni sui bus evitando che vi siano interferenze di dati.

Nel codice che controlla i motori, il cui flusso logico è mostrato sulla sinistra in Figura 62, il loop principale esegue solo il blocco "run motori" nel quale viene eseguita la logica di invio segnali ai driver sfruttando la libreria di arduino AccelStepper. Quando viene rilevato un segnale in ingresso, è eseguita la funzione che risolve la cinematica trasformando le velocità V_x e V_y nelle velocità ω_1, ω_2 e ω_3 delle ruote e viene impostata la velocità dei motori con la funzione SetSpeed presente nella libreria.

9 HARDWARE DI AZIONAMENTO E CONTROLLO

Al fine di facilitare i cablaggi e renderli più stabili si è deciso di creare delle schede mediante piastre millefori che presentassero dei connettori femmina attraverso i quali fosse possibile collegare le schede.

Figura 64: Schede Arduino e modulo GY-521 integrati su piastre forate

In Figura 64 sono mostrate entrambe le schede montate sulle piastre forate su cui sono state saldati opportunamente tutti i componenti necessari per il cablaggio e ulteriori elementi di supporto all'utilizzo del robot. Le schede e l'accelerometro presentano saldati direttamente i connettori maschi, pertanto sulle schede forate sono stati saldati i corrispettivi connettori femmina. I pin utilizzati per il collegamento logico con i motori, quelli di alimentazione delle schede e le linee di comunicazione I^2C, richiedono collegamenti via cavo, pertanto in corrispondenza degli stessi sono state aggiunte le morsettiere a vite.

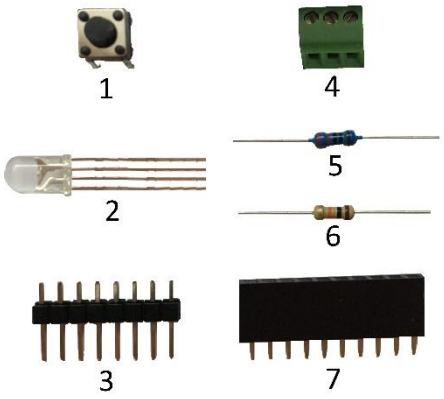


Figura 65: (1) Bottone, (2) Led RGB, (3) Connettore maschio, (4) Morsettiera, (5) Resistenza 220 ohm, (6) Resistenza 10k ohm, (7) Connettore femmina

In Figura 65 sono mostrati i componenti utilizzati nelle piastre forate. Il bottone (1) è utilizzato come pulsante ON/OFF per i motori, in questo modo è possibile, in fase di messa in moto del robot, azionare i motori in sicurezza ed eventualmente arrestarli. Il led RGB (2) è utilizzato come segnale di corretto funzionamento e inizializzazione della scheda GY-521, esso è saldato e collegato alla piastra che dell'Arduino MEGA 2560P. I componenti (3) e (7) mostrati in figura sono rispettivamente i connettori maschio e femmina utilizzati per il collegamento delle board con le piastre. L'elemento (4) è la morsettiera che presenta 3 pin.

Di seguito sono mostrate le piastre con i rispettivi schemi elettrici. I collegamenti mostrati negli schemi sono ottenuti sulle piastre mediante saldatura a stagno e spezzoni di filo.

9.1 Piastra forata Arduino UNO R3

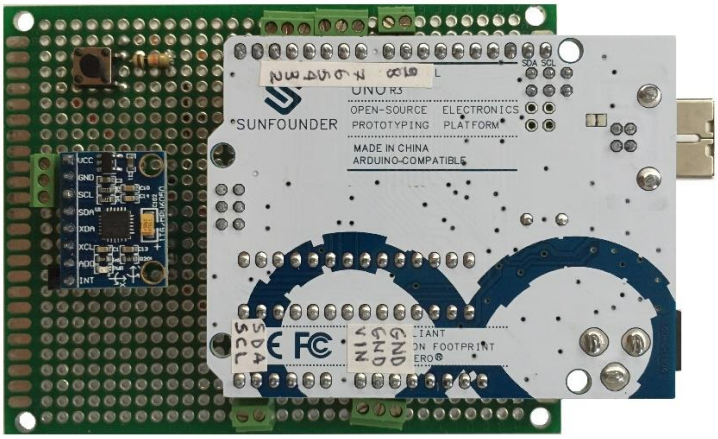


Figura 66: Piastra forata Arduino UNO R3

In Figura 66 è mostrata la piastra per intero con montate su di essa la scheda Arduino UNO R3 che rappresenta lo Slave dell'architettura di comunicazione e la scheda GY-521. Nella parte posteriore della scheda sono effettuati i collegamenti come mostrato in Figura 67.

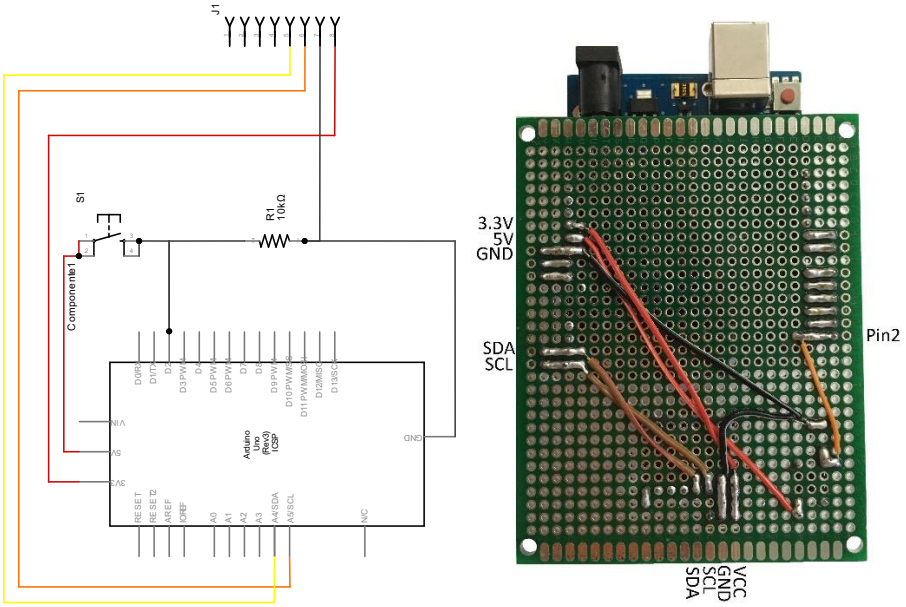


Figura 67: Schema di collegamenti Piastra forata Arduino UNO R3

L'accelerometro è alimentato mediante il pin 3.3V della scheda, mentre il pulsante è collegato al pin 5V; entrambi i circuiti vengono chiusi sul pin GND tramite il filo nero. Il filo arancione della figura di destra è il collegamento tra il pulsante e il pin 2.

I pin da 3 a 9, mediante i quali vengono controllati i driver, e i pin di alimentazione della scheda, sono saldati alla morsettiera.

9.2 Piastra forata Arduino MEGA 2560P

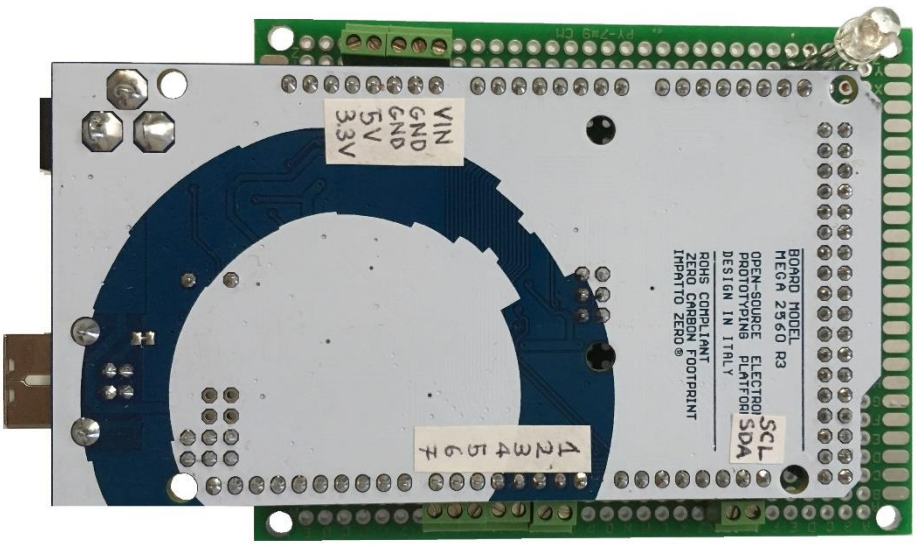


Figura 68: Piastra forata Arduino MEGA 2560

In Figura 68 è rappresentata la piastra forata, con le opportune saldature, su cui è montata la board Arduino MEGA 2560. La scheda in questione ha funzione di Master di comunicazione tra gli elementi, gestisce la scheda IMU leggendo i dati provenienti dall'accelerometro, legge i dati in ingresso dal controller[8] e invia le velocità dei motori ad Arduino UNO R3. Al fine di garantire un corretto funzionamento del protocollo di comunicazione I^2C sono stati saldati sulla piastra una morsettiera a due elementi in corrispondenza dei pin SDA e SCL. Ulteriori morsettiere sono collegate ai pin di alimentazione della scheda e ai pin di lettura del controller.

Alla piastra è stato aggiunto un led RGB, come segnale luminoso. Questo è risultato necessario poiché, in fase di funzionamento non è possibile leggere a video gli eventuali messaggi di errore provenienti dalla scheda.

[8] I dati in arrivo dal controller non fanno parte della linea di comunicazione comune agli elementi. La comunicazione avviene attraverso la lettura dei dati in arrivo sul pin digitale 3.

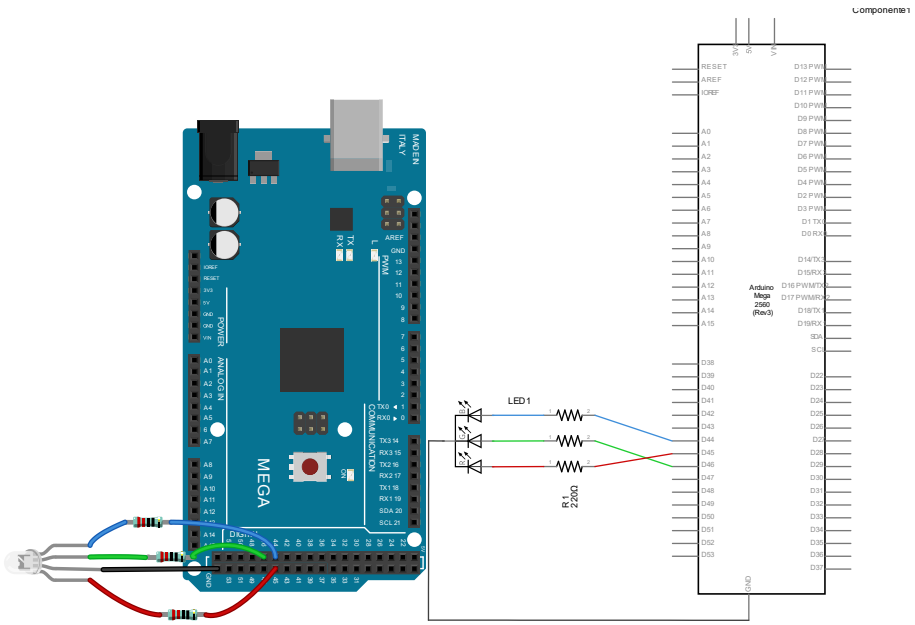


Figura 69: Collegamenti sulla piastra forata della scheda Arduinno MEGA 2560

In Figura 68 è rappresentato lo schema di collegamento effettuato sulla piastra mediante saldatura a stagno. Il led è alimentato dai pin 44, 45 e 46 tutti e tre in grado di inviare segnali PWM indispensabili per l'alimentazione del led.

Nell'algoritmo di lettura dei dati di inclinazione, è stato aggiunta una pausa del codice, in fase di setup del dispositivo, necessaria affinché la IMU abbia il tempo di stabilizzare la lettura. Un'eventuale lettura istantanea dei dati produrrebbe una lettura angolare imprecisa con conseguente mal funzionamento del controllo di stabilità.

Di seguito è mostrata la tabella dei segnali luminosi provenienti dal led.

Segnale	Significato
Azzurro lampeggiante	Segnala la corretta lettura della scheda GY-521 e lampeggia per un periodo di assestamento dei dati letti. Durante questa fase bisogna attivare il pulsante dei motori.
Rosso fisso	La scheda GY-521 non è stata letta correttamente, necessita il riavvio del dispositivo.
Verde fisso	Il dispositivo sta inviando correttamente i dati di velocità alla scheda Arduino UNO R3.
Spento	La lettura dei dati provenienti dalla GY-521 è interrotta da un eventuale interrupt sul pin 2.

Di seguito sono rappresentate le due piastre forate complete di saldature senza le schede montate.

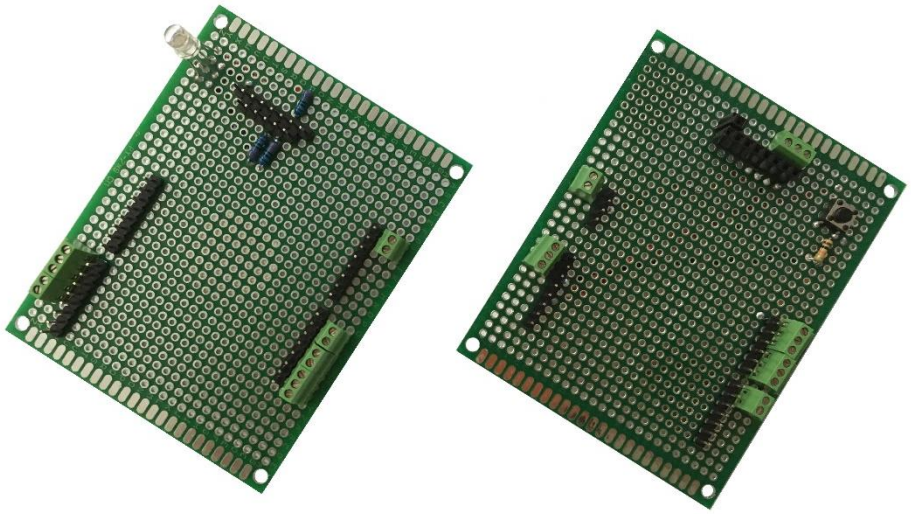


Figura 70: Piastre forate senza schede montate. Piastra di Arduino MEGA 2560 a sinistra, piastra di Arduino UNO R3 e modulo GY-521 a destra.

9.3 Test di cablaggio e collaudo parti

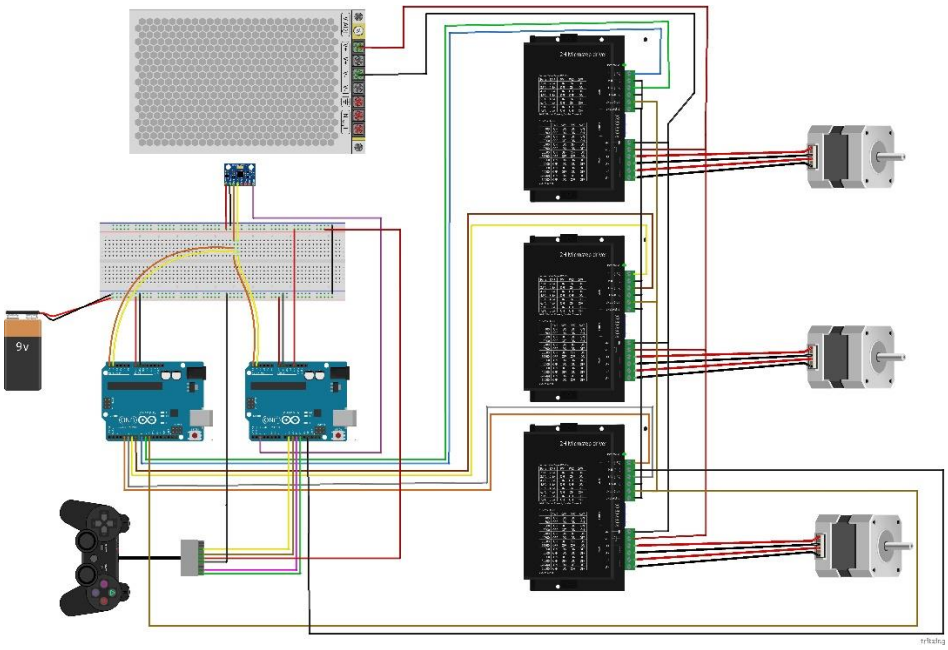


Figura 71: Collegamenti di test dei codici mediante breadboard

In Figura 71 sono mostrati i collegamenti effettuati in fase di test del codice effettuato per verificare il corretto funzionamento dei motori e del sistema di pilotaggio[9]. I driver sono alimentati da un alimentatore 24 VDC, i segnali di comando dei driver sono inviati dalla scheda di controllo dei motori (scheda a sinistra nell'immagine), la quale, eseguendo il codice mostrato in precedenza, manda un segnale a gradino ai driver. I pin relativi ai poli negativi dei driver, quali DIR-, PUL- e ENA-, devono essere collegati al pin GND delle board al fine di chiudere il circuito sui segnali di comando. Le linee di comunicazione tra le schede e l'accelerometro,

[9] Nello schema sono rappresentate due schede Arduino UNO, poiché inizialmente l'intenzione era di utilizzare queste schede, nel modello si è poi deciso di utilizzare una scheda Arduino MEGA2560 come già accennato in precedenza. Inoltre l'alimentazione delle schede, a differenza di quanto mostrato nello schema, non è stato affidato a una pila 9V collegata ai pin VIN ma a una power-bank che fornisce 5V in output collegata direttamente ai pin 5V. È importante sottolineare che l'alimentazione tramite i pin 5V deve essere effettuata mediante dispositivi che presentino una tensione in uscita di 5V stabilizzata. Un'eventuale picco di tensione in ingresso al pin 5V, brucerebbe irrimediabilmente la scheda. Questo è dovuto al fatto che il pin 5V delle schede Arduino bypassa il regolatore di tensione collegandosi direttamente al processore.

mostrate in giallo (SDA) e arancione (SCL) nel lato superiore delle schede, sono in comune per entrambe le comunicazioni.

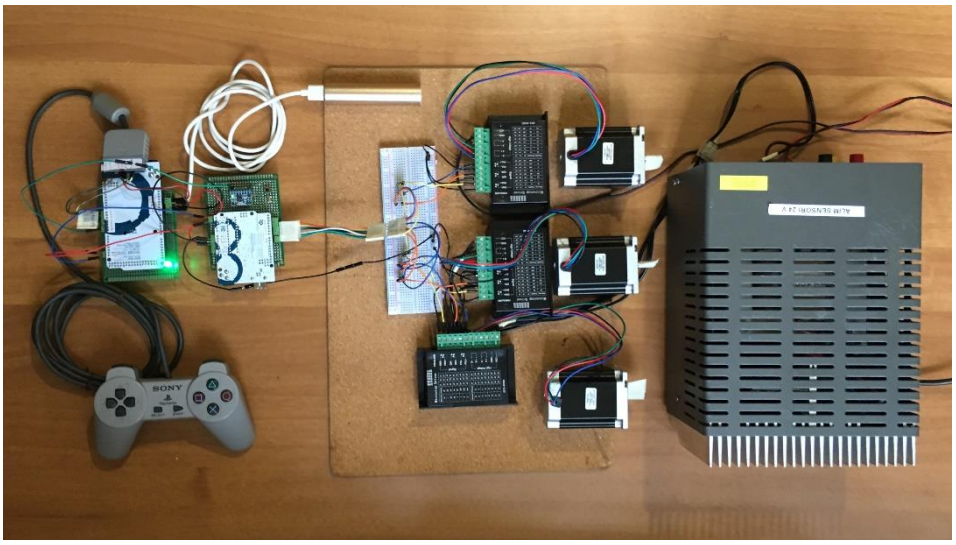


Figura 72: Test di collegamento delle parti su banco

SCHEDA Arduino UNO R3

	PIN
Motore 1:	
PUL+	3
DIR+	4
Motore 2:	
PUL+	5
DIR+	6
Motore 3:	
PUL+	7
DIR+	8
tutti i motori	
ENA+	9

SCHEDA Arduino MEGA 2650P

	PIN
Controller:	
Clock	4
Attent	5
Command	6
Data	7
GY-521	
Interrupt imu	2

I pin SDA e SCL di entrambe le schede devono essere collegati assieme. Per quanto riguarda l'alimentazione vanno collegati i pin 5V di entrambe le schede e i pin GND in comune con l'alimentazione che deve essere stabilizzata e mai superiore a 5V.

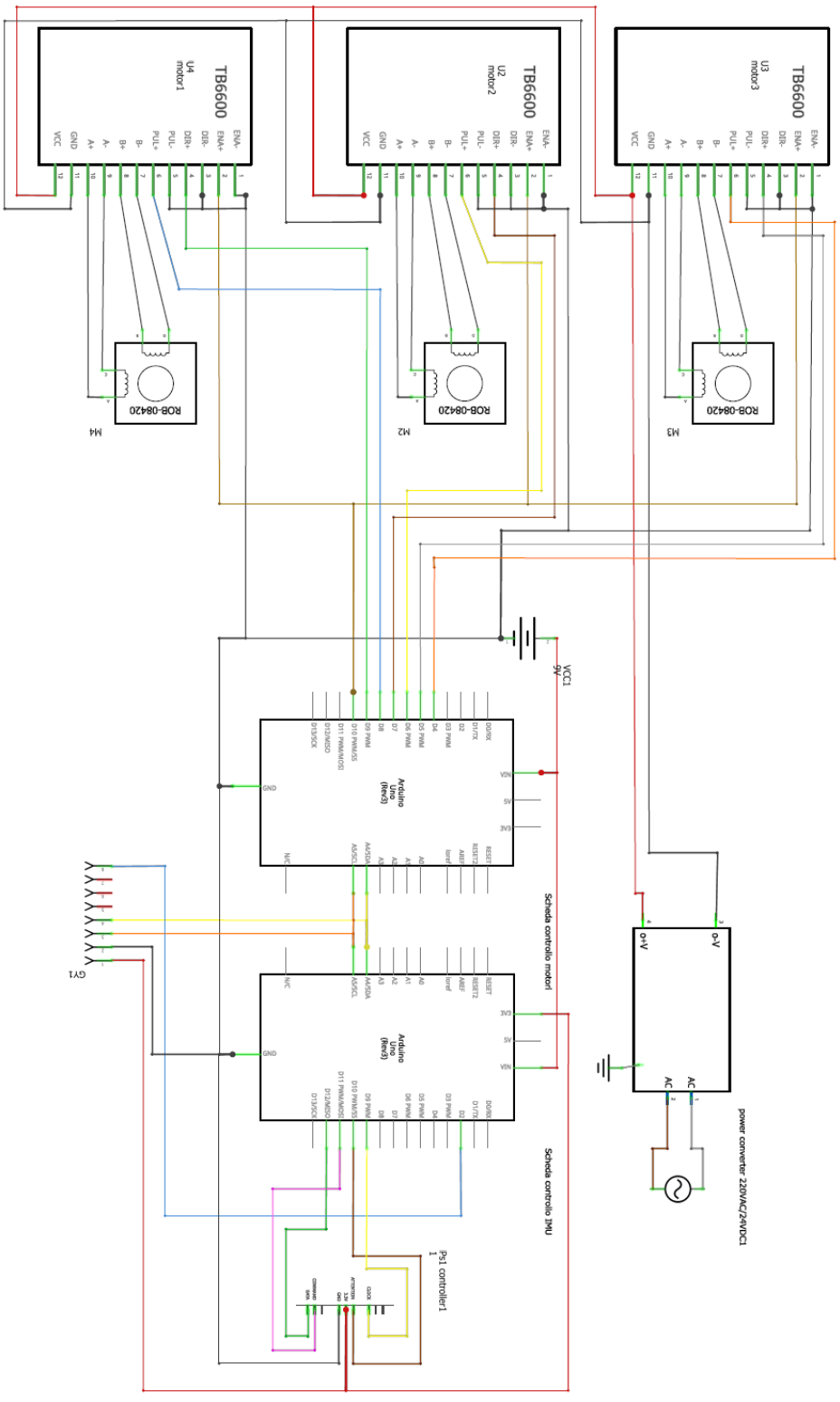


Figura 73: Schema cablaggi

10 ASSEMBLAGGIO E PRIMI TEST

In questo capitolo si analizza la parte di assemblaggio delle componenti del robot con l'ausilio di alcune viste esplose dell'assieme. Dopo essere stato assemblato, prima di effettuare il test di equilibrio e procedere con il tuning del controllo, si è preferito testare gli algoritmi cinematici inseriti all'interno del software per il controllo dei motori. Dopo aver testato la cinematica è stata fatta la calibrazione dell'accelerometro in modo da fornire in input gli offset di lettura sull'inclinazione. Avendo terminato la calibrazione si è proceduto con la messa a punto del controllo. Dopo aver raggiunto un buon risultato di stabilità e movimento sono stati fatti alcuni test sperimentali utili per mettere a punto il controllo in maniera più approfondita analizzando quantitativamente il comportamento del robot. Di seguito sono mostrati nel dettagli i passi sopra descritti.

10.1 Assemblaggio parti

Sono stati prima montate le flange sulle ruote mediante viti con testa scavata a brugola M5x40, dopodiché sono stati montati i motori sui supporti mediante viti a testa esagonale M5x20. Infine si è proceduto montando le ruote sui motori fissandole con l'utilizzo di un grano M3x14.

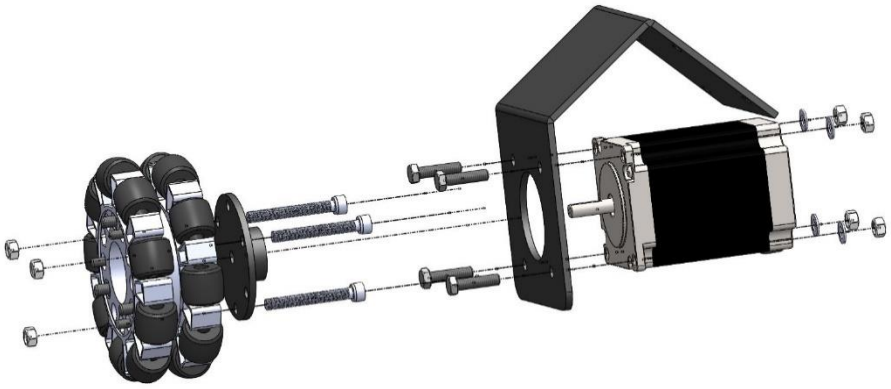


Figura 74: Vista esplosa del montaggio del blocco motore

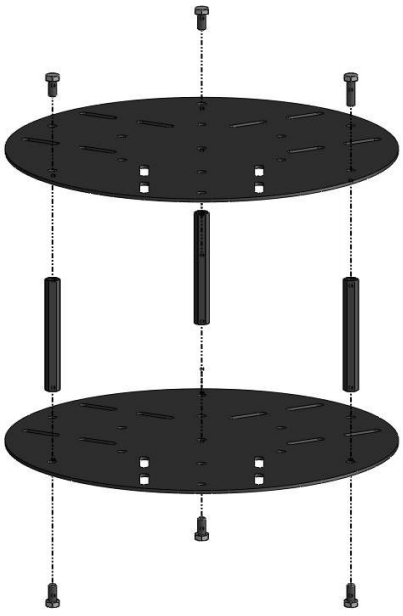


Figura 75: Vista esplosa delle piastre superiori

In questo modo si è ottenuto il blocco motore pronto ad essere montato sulle piastre di supporto .Le piastre superiori sono state montate mediante l'utilizzo di distanziali M5x70 e viti a testa esagonale M5x10 come visibile in Figura 75. Il montaggio della piastra più in alto è avvenuto dopo aver effettuato i cablaggi dei driver per facilitarne le operazioni. Infine sono stati montati i blocchi motori sul blocco piastre ottenendo così il robot completo. Le schede forate sono state fissate alla piastra superiore sfruttando le asole già presenti e sagomando opportunamente le piastre.

10.2 Prova sperimentale di cinematica

Figura 76: Test cinematico

Dopo aver assemblato il robot sono stati effettuati alcuni test preliminari al fine di verificare il corretto funzionamento cinematico dell'attuazione. È stato rovesciato il robot ed è stata posizionata la palla sopra di esso come visibile in Figura 76. In questa posizione sono state fornite in input le velocità lungo X e lungo Y. Si è verificato che le ruote trasferissero alla sfera la rotazione tale da imporre uno spostamento del robot lungo l'asse corrispondente.

Dal test è emerso anche che l'attrito tra ruote e sfera è sufficiente a garantire un buon funzionamento nonostante la forza di contatto in quel caso fosse determinata solo dal peso della sfera. Nei test successivi è stato poi confermato il risultato emerso dal primo test cinematico.

Dopo aver verificato che l'algoritmo cinematico fosse corretto, sono state effettuate alcune prove con lo scopo di mettere a punto il controllo sia di stabilità che di posizione nello spazio.

10.3 Tuning del controllo

Figura 77: Ballbot in configurazione standard per prove di controllo

Per effettuare la messa a punto del controllo è stato inizialmente disattivato il controllo sulla posizione lasciando attivo solo quello di equilibrio. Partendo con dei dati ragionevoli di guadagno proporzionale, ricavati dal modello Simulink, è stato aumentato il valore fino a raggiungere un valore per il quale il robot oscillasse stabilmente attorno alla posizione di equilibrio. Dopo aver ricavato tale valore, si è proceduto aumentando il guadagno derivativo fino al raggiungimento di un comportamento stabile del robot.

In seguito è stata fatta in maniera analoga la messa a punto del controllo di posizione. Durante questo test si è notato che un valore troppo basso del guadagno proporzionale del

non producesse alcuna traslazione del robot in quanto il robot ristabilisse la posizione di equilibrio prima di aver cominciato la traslazione. Pertanto, dopo aver raggiunto il primo valore per cui il robot cominciasse a traslare, si è proceduto a diminuire l'overshoot sulla posizione aumentando il guadagno derivativo.

#	1	2	3	4	5	6	7	8	9	10	11	12
Ka	10	12.5	14.5	15	15	15	15	15	15	15	15	**15**
Kav	1	1	1	1	2	3	2.2	2.2	2.2	2.2	2.2	**2.2**
Kt	0	0	0	0	0	0	0.4	0.8	1	1	1	**1**
Kv	0	0	0	0	0	0	0	0	0	0.5	1.5	**2**

I guadagni ricavati sperimentalmente consentono al robot di mantenere la posizione di equilibrio e di muoversi secondo i comandi di input, riuscendo a gestire anche un input a gradino sulla posizione e forze sbilancianti di disturbo.

10.4 Prove sperimentali di dinamica

Sono stati raccolti i dati di alcune prove sperimentali di equilibrio e spostamento del robot al fine di analizzarne in maniera più quantitativa il comportamento.

Prova 1: equilibrio statico in assenza di disturbo

In questa prova il robot deve mantenere la sua posizione iniziale e mantenersi in equilibrio.

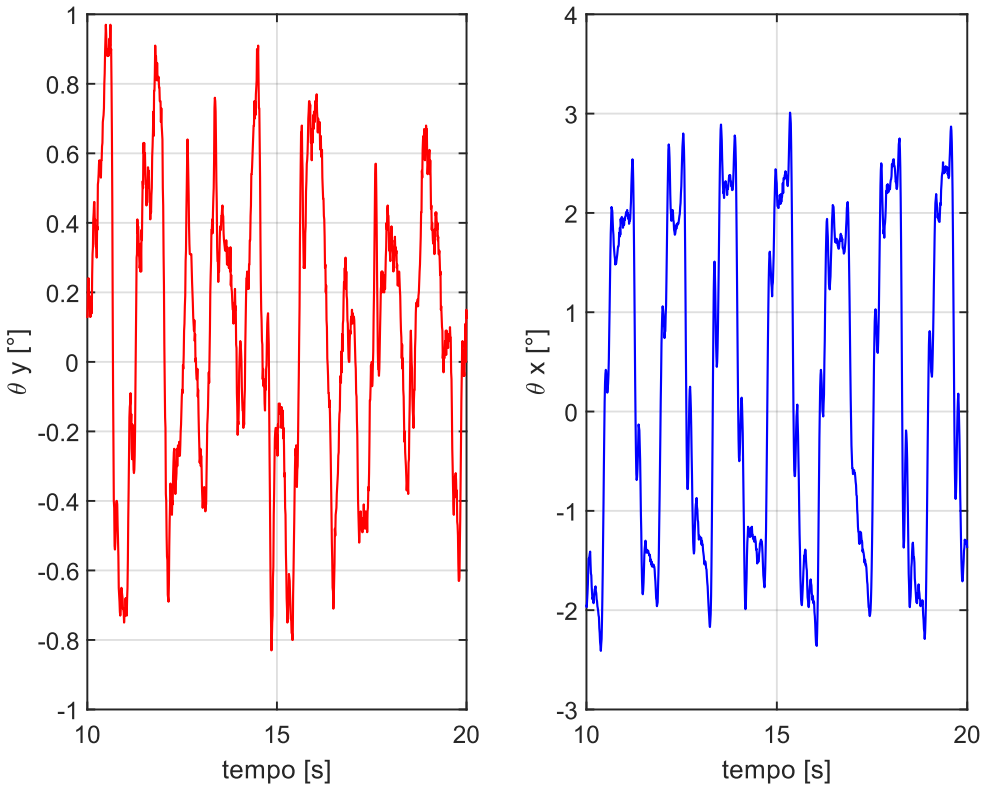


Figura 78: Angoli di inclinazione prova1 di stabilità

Si noti come il robot continui a oscillare attorno alla posizione di equilibrio in entrambe le direzioni. In Figura 78 si evince come l'inclinazione attorno all'asse X vari circa tra ± 3 gradi mentre attorno all'asse y sia circa un terzo. Questo è dovuto al fatto che l'accelerometro, fissato solo con l'ausilio dei pin, ha un piccolo gioco di posizionamento, pertanto spostando il robot

prima della prova è stato mosso inavvertitamente introducendo un errore di lettura dell'angolo di inclinazione. Questo ha causato un errore di lettura e la posizione di equilibrio cercata dal robot non corrispondeva all'effettiva posizione di equilibrio, causando un'oscillazione più ampia.

Considerazione analoga può essere fatta analizzando la posizione calcolata come integrale delle velocità imposte alla sfera dall'attuazione.

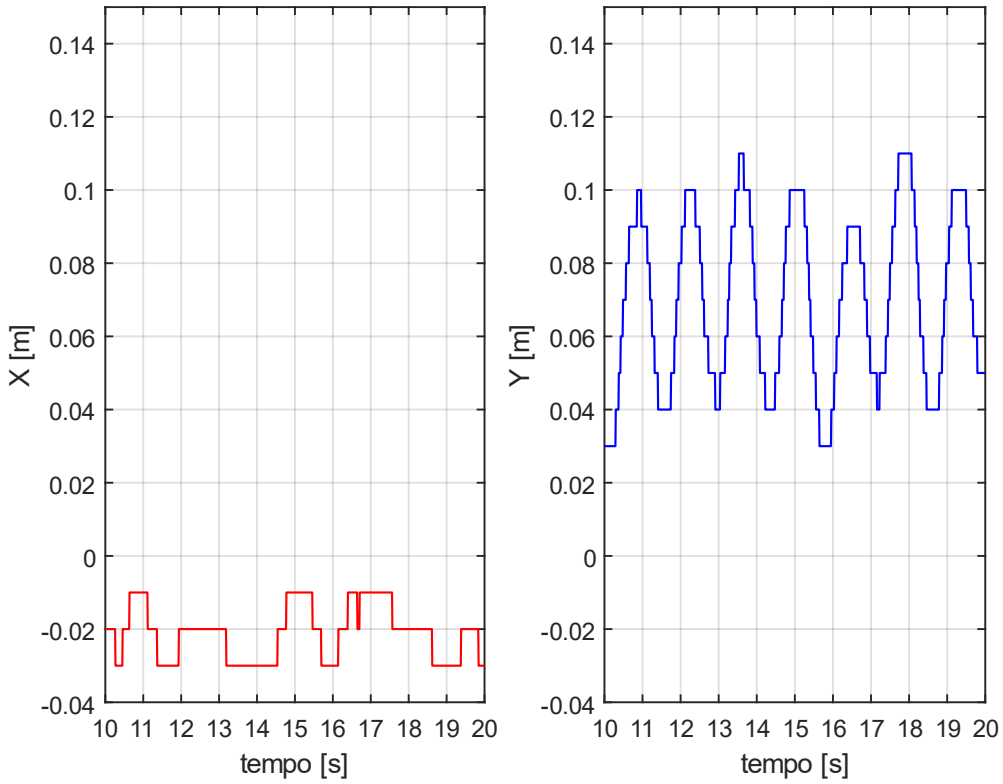


Figura 79: Posizione prova1 di stabilità

Anche in questo grafico emerge che la traslazione lungo Y, quindi causata da una rotazione della sfera attorno all'asse X, escludendo l'errore a regime di 0.07 m, oscilli con un'ampiezza di 3 cm, mentre la posizione in direzione X oscilli con un'ampiezza di 1 cm.

La presenza dell'errore a regime è coerente con il controllo di tipo PD, in quanto non è stato aggiunto un guadagno integrativo sulla posizione che diminuirebbe l'errore a regime.

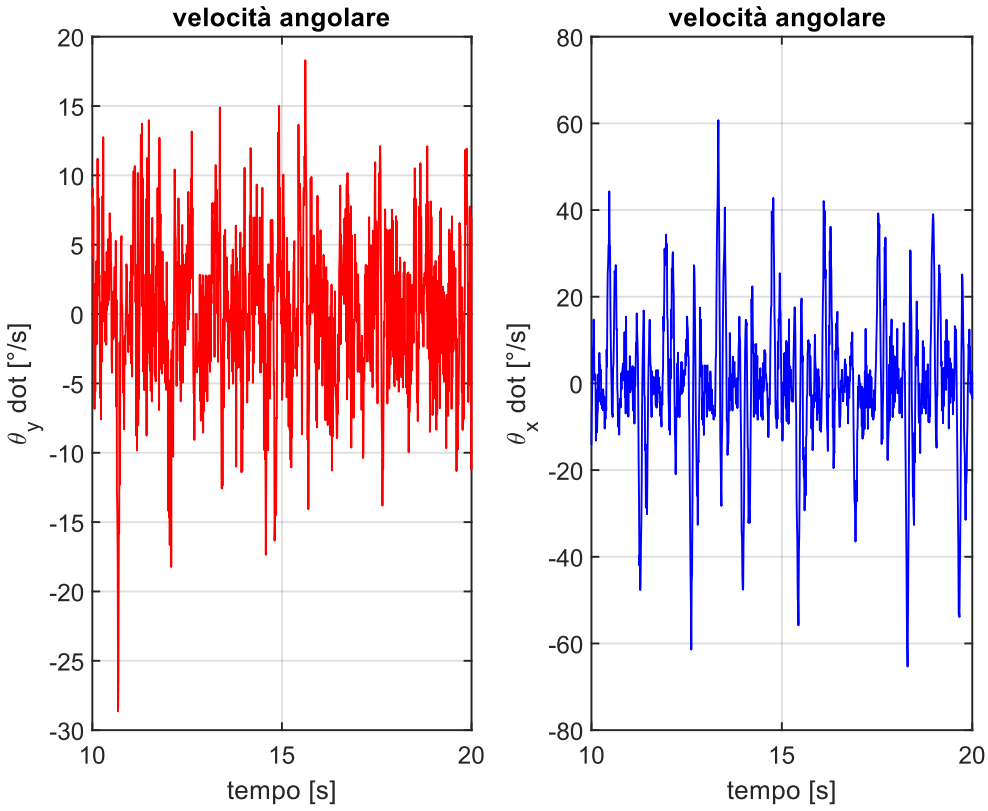


Figura 80: Velocità angolari prova1 di stabilità

Dal grafico delle velocità angolari si nota quanto questa grandezza sia affetta da rumore, per ottenere un risultato più pulito sarebbe opportuno inserire un filtro nel codice per ripulire i segnali dal rumore.

Prova 2: equilibrio statico in assenza di disturbo

È stato rieseguito il test dopo aver smontato e rimontato la scheda GY-521 cercando di rimontarla al meglio. Di seguito sono riportati i risultati ottenuti in termini di inclinazione attorno agli assi X e Y.

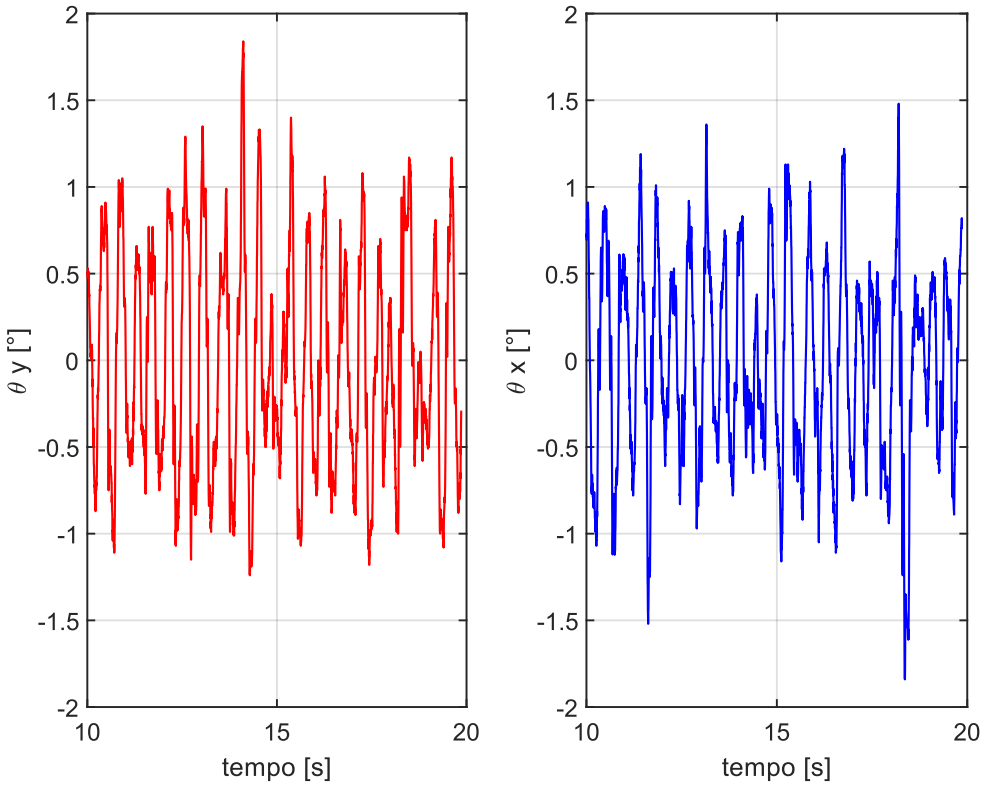


Figura 81: Angoli di inclinazione prova2 di stabilità

Si noti come l'ampiezza di oscillazione attorno all'asse X si sia ridotta rispetto alla prima prova. È necessario al fine di ottenere segnali di inclinazione privi di errori aggiungere al prototipo un sistema di fissaggio del sensore più stabile in modo tale da evitare eventuali spostamenti durante l'utilizzo. Così facendo, dopo aver effettuato la calibrazione preliminare in fase di montaggio per compensare eventuali errori, la lettura risulterà corretta anche in caso di urti della scheda con altre componenti.

Prova 3: equilibrio statico con forza di disturbo in direzione X

È stata effettuata una prova di stabilità durante la quale, spingendo il robot dalla piastra superiore in direzione X, è stata introdotta una forza di disturbo.

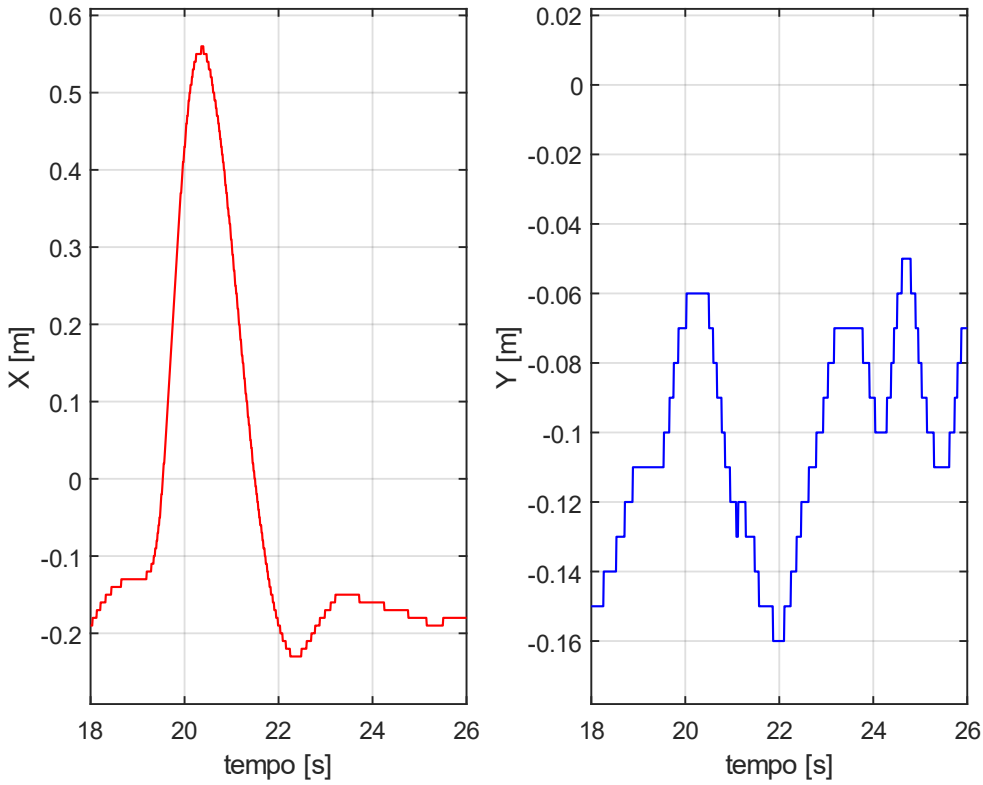


Figura 82: Posizione prova1 di stabilità con disturbo

Come si può vedere in Figura 82, in corrispondenza tempo 19 sulle ascisse, è stato introdotto il disturbo in direzione X. Il robot ha effettuata una traslazione per compensare l'inclinazione introdotta dal disturbo per poi ritornare nella posizione iniziale[10].

[10] Si noti che la posizione iniziale in questo caso corrisponda al valore -0.2 m. Questo è dovuto al fatto che prima del disturbo rappresentato in figura erano stati introdotti altri disturbi consecutivi, per questo la posizione lungo l'asse X, così come anche sull'asse Y ha accumulato l'errore a regime causato dall'assenza di un guadagno integrativo nel controllo.

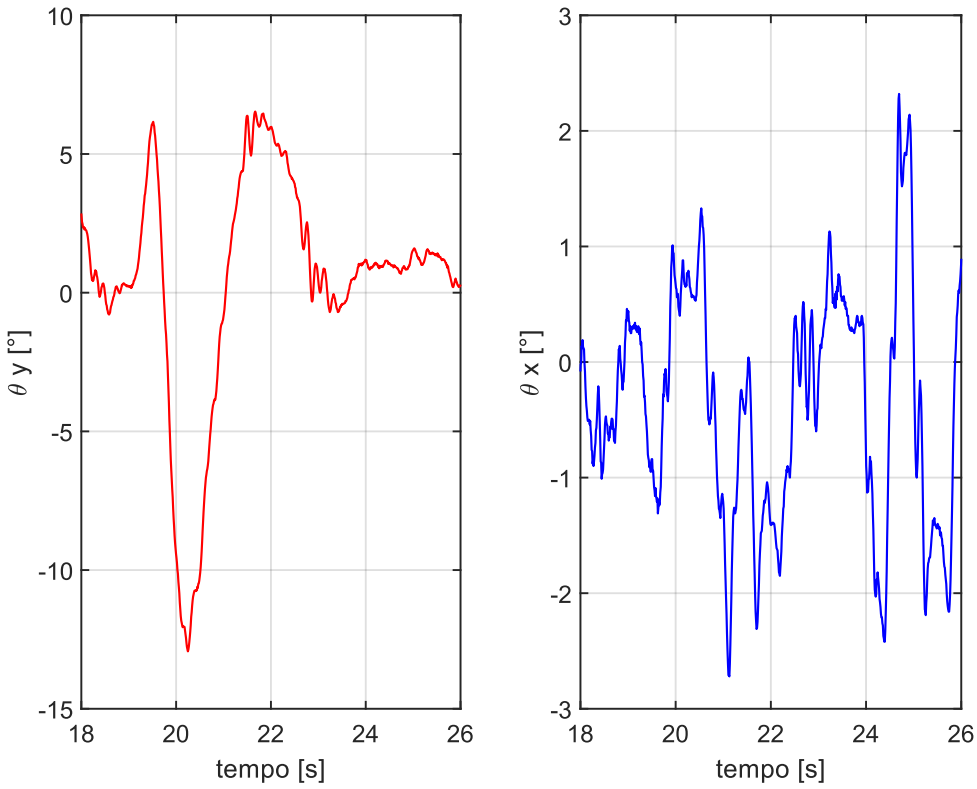


Figura 83: Angoli di inclinazione prova3 stabilità con disturbo

In Figura 83 sono rappresentati gli angoli di inclinazione durante la reazione del robot al disturbo. Per analizzare nel dettaglio il comportamento, già analizzato con il modello simulink, si ricorre ai grafici in Figura 84.

Di seguito si analizzano cronologicamente i tratti in ascissa della prova 1 in riferimento al Figura 84.

19-19.5

Il ballbot è sbilanciato con un angolo di inclinazione ϑ_y positivo, il controllo impone un'accelerazione positiva alla sfera da cui risulta una velocità V_x positiva e di conseguenza una traslazione nella stessa direzione.

19.5-20.2

L'accelerazione imposta alla sfera produce una diminuzione dell'angolo di inclinazione raggiungendo valori negativi di inclinazione per poter successivamente recuperare l'equilibrio ritornando verso la posizione iniziale. In questo periodo il robot inizia una fase di decelerazione della sfera fino a raggiungere la massima inclinazione e velocità nulla. In 20.2 il robot ha massima inclinazione negativa e massimo spostamento lungo X.

20.2-21

In questa fase continua l'accelerazione negativa quasi costante fino a raggiungere la massima velocità di richiamo in posizione di partenza. Essendosi sbilanciato in precedenza con inclinazione negativa, in questa fase di accelerazione negativa, si recupera sia la posizione sia l'inclinazione.

21-24

durante la traslazione per tornare nella posizione iniziale è presente un piccolo overshoot nel controllo inclinazione che risulta necessario per limitare l'overshoot nel controllo posizione pari a qualche centimetro in questa prova.

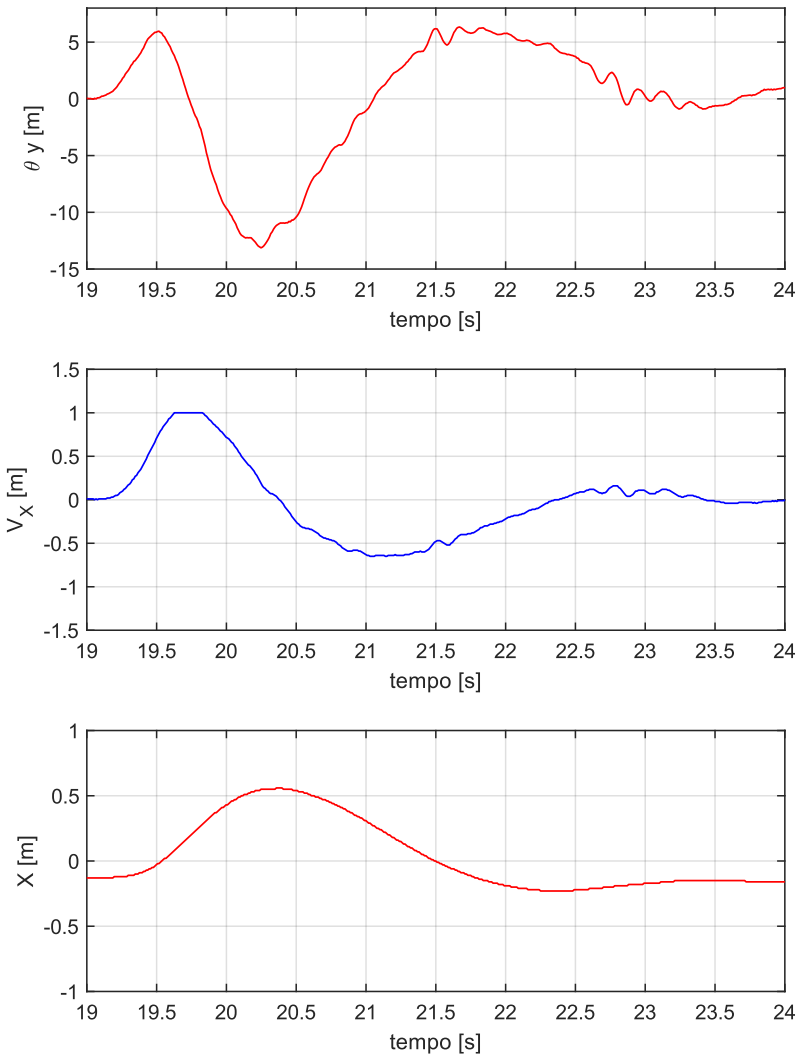


Figura 84: Grafici di prova1 di stabilità con disturbo

Di seguito si riportano i grafici analoghi di ulteriori prove effettuate di stabilità con disturbo di forza.

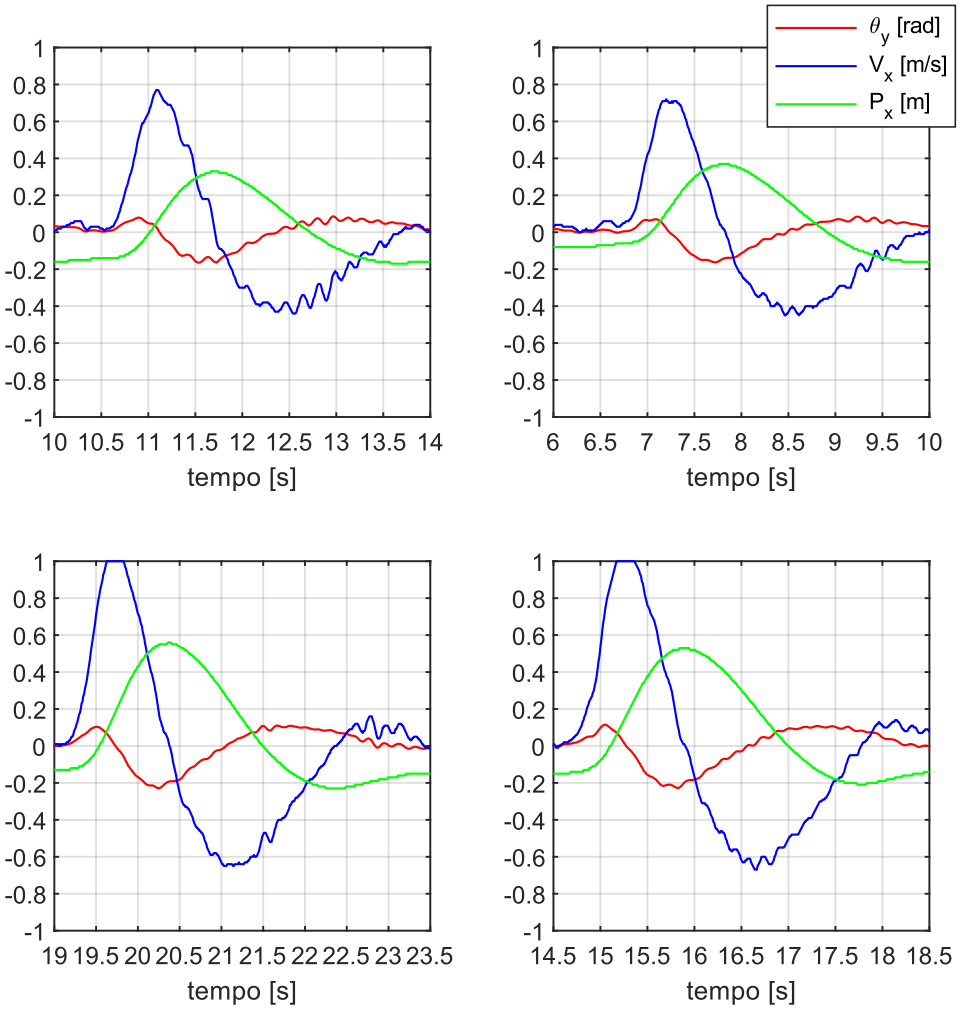


Figura 85: Prove di stabilità con disturbo lungo X

Prova 4: ingresso a gradino sulla posizione

In questa prova è fornito in input un ingresso a velocità molto alta che può essere considerato pari ad un ingresso a gradino.

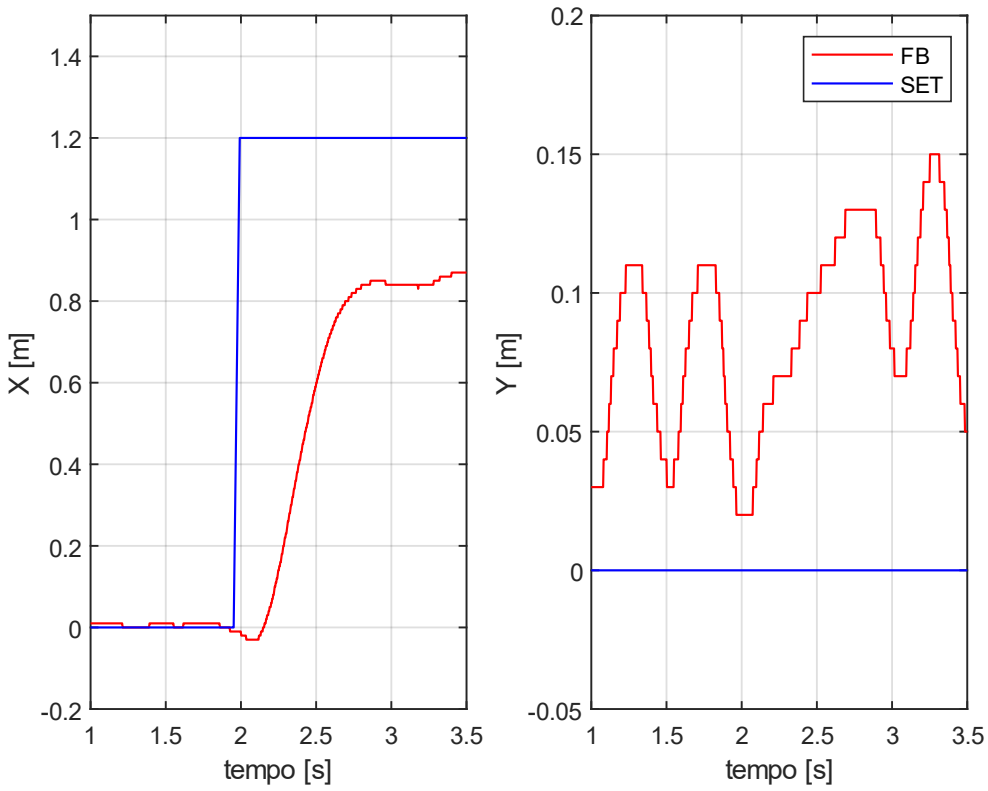


Figura 86: posizione SET e FB per un ingrasso a gradino

Il comando di SET è pari a 1.2 m. Il controllo proporzionale della posizione agisce con segno negativo rispetto gli altri guadagni quindi un errore di posizione (SET-FB) positivo impone alla sfera un'accelerazione negativa. Come si vede al tempo 2 di Figura 86 a sinistra, questa accelerazione produce una traslazione della sfera in direzione opposta a quella del SET. Subendo una brusca accelerazione il robot si sbilancia vistosamente nella direzione del comando di SET (come visibile in Figura 87 per l'angolo ϑ_y) pertanto il controllo sull'inclinazione, che è circa di un ordine di grandezza superiore a quello della posizione impone l'accelerazione in direzione X e il ballbot recupera l'inclinazione muovendosi nella direzione

desiderata. Tuttavia, una volta raggiunto l'equilibrio in prossimità del SET, il robot non riesce ad eliminare l'errore a regime. Questo è coerente con il controllo implementato di tipo PD, il quale non presentando una componente integrativa nel controllo posizione non compensa l'errore a regime.

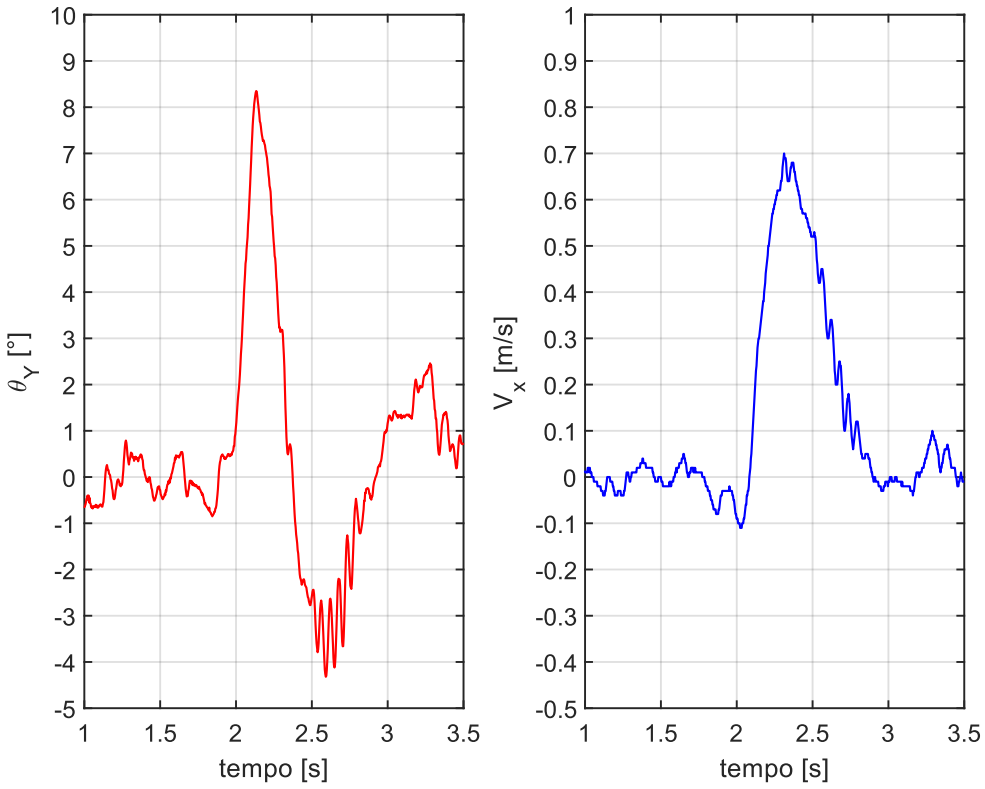


Figura 87: inclinazione e velocità durante la prova con ingresso a gradino sulla posizione

Di seguito sono riportate ulteriori prove effettuate con ingresso a gradino sulla posizione

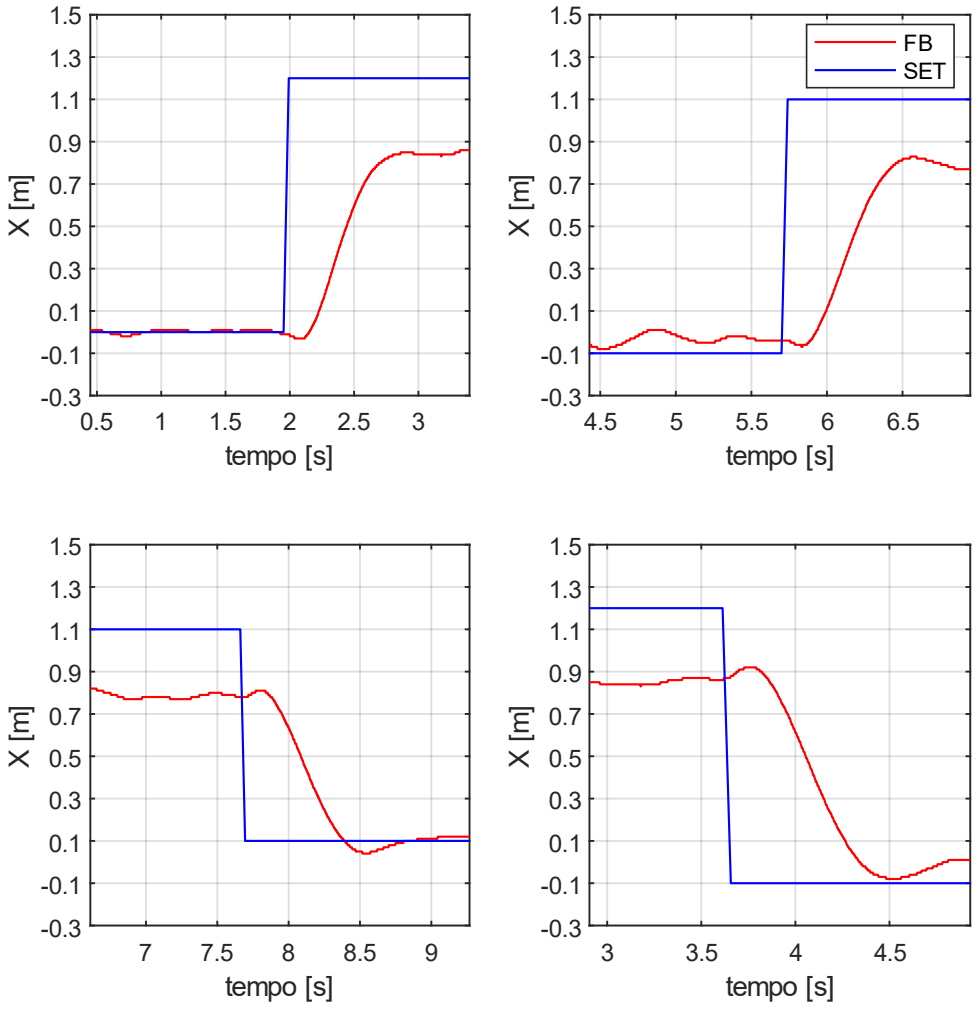


Figura 88: Prove di traslazione con ingresso a gradino lungo X

Prova 5: ingresso a rampa sulla posizione

È stata condotta anche una prova, analoga alla precedente, di traslazione ma fornendo in input una rampa sulla posizione, ovvero una velocità di traslazione costante lungo X.

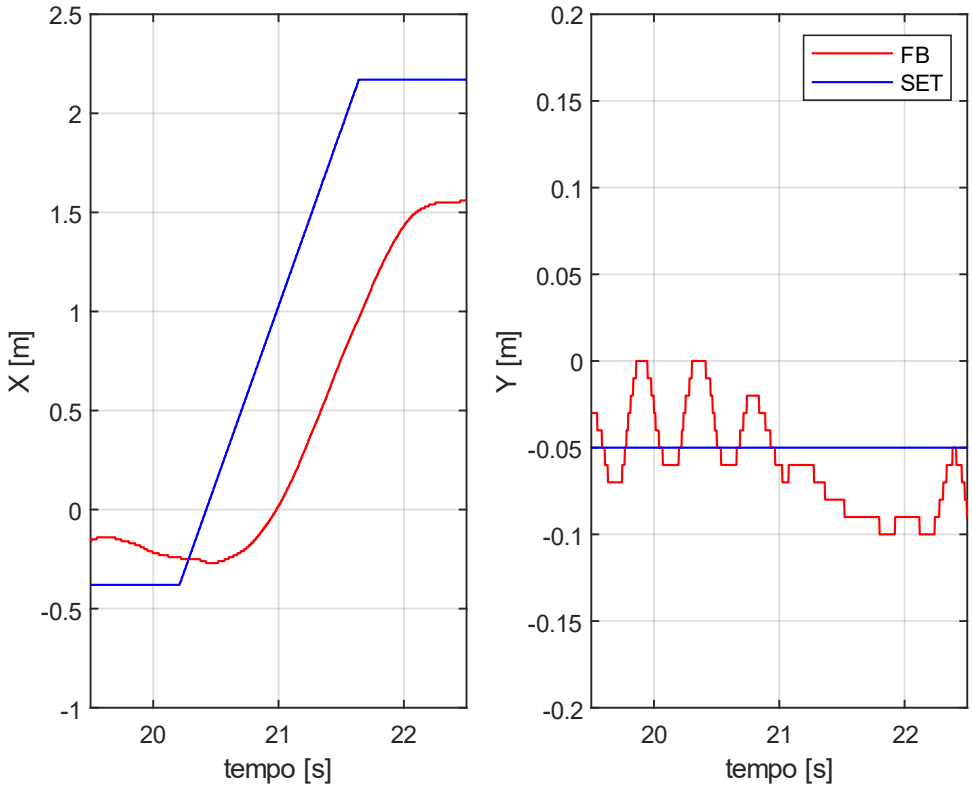


Figura 89:posizione SET e FB per un ingrasso a rampa

Anche in questo caso, il robot si muove con un certo errore a regime sia durante il profilo di velocità che nella posizione finale, sempre coerentemente con il tipo di controllo adottato.

APPENDICE 1

Sono riportati in questa appendice i codici caricati sulle due schede Arduino e gli script Matlab adottati.

Codice Slave:

```
#include <AccelStepper.h>
#include <Wire.h>
#include <math.h>
#include <I2C_Anything.h>

#define sinfi sin(fi)//0.6428 // utilizzando fi 40°
#define cosfi cos(fi)//0.766 // utilizzando fi 40°

volatile boolean haveData = false;
volatile float Vx=0;
volatile float Vy=0;
const int buttonPin=2; //Pin di lettura segnale pulsante
int step1 = 8;      //Step motore 1 sul pin 8
int dir1 = 9;       //dir motore 1 sul pin 9
int step2 = 6;      //Step motore 2 sul pin 6
int dir2 = 7;       //dir motore 2 sul pin 7
int step3 = 4;      //Step motore 3 sul pin 4
int dir3 = 5;       //dir motore 3 sul pin 5
int en1 = 3;       //Ena+ dei 3 motori sul pin 2
float fi=0.07071;
float r=0.05;
float w1=0;
float w2=0;
float w3=0;
float passo=0.25; // 1 1/2 1/4 1/8 1/6
float rad2passo=200/(passo*2*M_PI);  //trasforma rad/s in passi/s
bool B= false;
bool p=false;

AccelStepper stepper1(1, step1, dir1);
AccelStepper stepper2(1, step2, dir2);
AccelStepper stepper3(1, step3, dir3);

 void setup() {

 pinMode(step1, OUTPUT);
 pinMode(dir1, OUTPUT);
 pinMode(step2, OUTPUT);
 pinMode(dir2, OUTPUT);
 pinMode(step3, OUTPUT);
 pinMode(dir3, OUTPUT);
pinMode(buttonPin, INPUT);
 digitalWrite(en1,LOW);

 Wire.begin (4);
```

```
  Wire.onReceive (receiveEvent);

  stepper1.setMaxSpeed(800); //step/seconds
  stepper2.setMaxSpeed(800); //step/seconds
  stepper3.setMaxSpeed(800); //step/seconds
}
//_____________LOOP PROGRAM_____________//
void loop(){

        buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH){
    p=!B;
    B=p;
    delay(1000);
    }

if (B==true){
if (haveData)
    {
          cinematica();
          //plot();
          stepper1.setSpeed(w1*rad2passo);  //step/seconds
          stepper2.setSpeed(w2*rad2passo);  //step/seconds
          stepper3.setSpeed(w3*rad2passo);  //step/seconds

          haveData = false;
  }  // end if haveData


stepper1.runSpeed();
stepper2.runSpeed();
stepper3.runSpeed();
}
            }

 void receiveEvent (int howMany)
 {
   I2C_readAnything(Vx);        //m/s
   I2C_readAnything(Vy);        //m/s
   haveData = true;
   // end if have enough data
 }  // end

 void cinematica(){

  w1=(-cosfi*Vy)/r;                          //rad/s
  w2=(0.866*cosfi*Vx+0.5*cosfi*Vy)/r;    //rad/s
  w3=(-0.866*cosfi*Vx+0.5*cosfi*Vy)/r;   //rad/s
  }
```

Codice Master

```
#include "I2Cdev.h"
#include <TimerOne.h>
#include <I2C_Anything.h>
#include <Psx.h>
#include "MPU6050_6Axis_MotionApps20.h"
#include <math.h>

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for AD0 high

/* =========================================================
   NOTE: In addition to connection 3.3v, GND, SDA, and SCL, this sketch
   depends on the MPU-6050's INT pin being connected to the Arduino's
   external interrupt #0 pin. On the Arduino Uno and Mega 2560, this is
   digital I/O pin 2.
 * ========================================================= */

/* =========================================================
   NOTE: Arduino v1.0.1 with the Leonardo board generates a compile error
   when using Serial.write(buf, len). The Teapot output uses this method.
   The solution requires a modification to the Arduino USBAPI.h file, which
   is fortunately simple, but annoying. This will be fixed in the next IDE
   release. For more info, see these links:

   http://arduino.cc/forum/index.php/topic,109987.0.html
   http://code.google.com/p/arduino/issues/detail?id=958
 * ========================================================== */

#define dataPin 12
#define cmndPin 11
#define attPin 10
#define clockPin 9
#define X 512
#define O 1024
#define quadrato 256
#define triangolo 2048
```

```
#define su 8
#define giu 2
#define dx 4
#define sx 1
#define R1 4096
#define R2 16384
#define L1 8192
#define L2 32768
#define vel 0.6
Psx Psx;                                // Inizializzazione libreria

unsigned int data = 0;                  // data stores the controller response

// uncomment "OUTPUT_READABLE_YAWPITCHROLL" if you want to see the yaw/
// pitch/roll angles (in degrees) calculated from the quaternions coming
// from the FIFO. Note this also requires gravity vector calculations.
// Also note that yaw/pitch/roll angles suffer from gimbal lock (for
// more info, see: http://en.wikipedia.org/wiki/Gimbal_lock)
#define OUTPUT_READABLE_YAWPITCHROLL


#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most boards
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

// packet structure for InvenSense teapot demo
//uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };
long temp1=0;
long temp2=0;
long tt=0;
int dt=0;

float yaw1=0;
float pitch1=0;
float roll1=0;
float yaw_vel=0;
float pitch_vel=0;
float roll_vel=0;
```

```
int ka=50;  //guadagno inclinazione
int kav=0; //guadagno velocità inclinazione
int kt=6;   //guadagno posizione
int ki=0;   //guadagno integrativo posizione
int kv=7;   //guadagno derivativo posizione (velocità)
int vmax=1;

float Ax=0;
float Ay=0;
float Vx=0;
float Vy=0;
float Px=0;
float Py=0;
float Pxset=0; //m
float Pyset=0; //m

//============================================================
// ===              INTERRUPT DETECTION ROUTINE               ===
//============================================================

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
  mpuInterrupt = true;
}

//============================================================
// ===                  INITIAL SETUP                     ===
//============================================================

void setup() {
  Psx.setupPins(dataPin, cmndPin, attPin, clockPin, 10);  // Defines what each pin is used
  // join I2C bus (I2Cdev library doesn't do this automatically)
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
    Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
  #endif

  // initialize serial communication
  // (115200 chosen because it is required for Teapot Demo output, but it's
  // really up to you depending on your project)
  Serial.begin(115200);
  while (!Serial); // wait for Leonardo enumeration, others continue immediately

  // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3V or Arduino
  // Pro Mini running at 3.3V, cannot handle this baud rate reliably due to
  // the baud timing being too misaligned with processor ticks. You must use
  // 38400 or slower in these cases, or use some kind of external separate
  // crystal solution for the UART timer.

  // initialize device
  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();
  pinMode(INTERRUPT_PIN, INPUT);
```

```
    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    // make sure it worked (returns 0 if so)
    if (devStatus == 0) {
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // enable Arduino interrupt detection
        Serial.print(F("Enabling interrupt detection (Arduino external interrupt "));
        Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
        Serial.println(F(")..."));
        attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        // set our DMP Ready flag so the main loop() function knows it's okay to use it
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;

        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}
// ================================================================
// ===                    MAIN PROGRAM LOOP                     ===
// ================================================================
void loop() {
    data = Psx.read();                          // Psx.read() initiates the PSX controller and returns

     // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize) {
        if (mpuInterrupt && fifoCount < packetSize) {
          // try to get out of the infinite loop
          fifoCount = mpu.getFIFOCount();
        }
    }
```

```
    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();
    // check for overflow (this should never happen unless our code is too inefficient)
    if ((mpuIntStatus & _BV(MPU6050_INTERRUPT_FIFO_OFLOW_BIT)) || fifoCount >= 1024) {
       // reset so we can continue cleanly
       mpu.resetFIFO();
       Serial.println(F("FIFO overflow!"));

    // otherwise, check for DMP data ready interrupt (this should happen frequently)
    } else if (mpuIntStatus & _BV(MPU6050_INTERRUPT_DMP_INT_BIT)) {

       // read a packet from FIFO
  while(fifoCount >= packetSize){ // Lets catch up to NOW, someone is using the dreaded delay()!
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    // track FIFO count here in case there is > 1 packet available
    // (this lets us immediately read more without waiting for an interrupt)
    fifoCount -= packetSize;
  }

     comando(data,dt);        // lettura del comando di set



       #ifdef OUTPUT_READABLE_YAWPITCHROLL
          // display Euler angles in degrees
          mpu.dmpGetQuaternion(&q, fifoBuffer);
          mpu.dmpGetGravity(&gravity, &q);
          mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
       #endif

         temp2=micros();
         dt=temp2-temp1;
         temp1=micros();


       // blink LED to indicate activity
       blinkState = !blinkState;
       digitalWrite(LED_PIN, blinkState);


yaw_vel=(ypr[0]-yaw1)*1000000/(dt);
pitch_vel=(ypr[1]-pitch1)*1000000/(dt);
roll_vel=(ypr[2]-roll1)*1000000/(dt);


yaw1=ypr[0];
pitch1=ypr[1];
roll1=ypr[2];


controllo();
```

```
   if (temp1-tt>100000){
        manda_dati();
        tt=temp1;
    }
  }
}


void manda_dati(){
   Wire.beginTransmission (4);
   I2C_writeAnything (Vx);
   I2C_writeAnything (Vy);
   Wire.endTransmission ();
}


void controllo(){

// piano XZ
Ax=-(ypr[1]*ka+pitch_vel*kav-kt*(Pxset-Px)+kv*Vx);
Vx=Vx+Ax*dt/1000000;
Px=Px+Vx*dt/1000000;

// piano YZ
Ay=(ypr[2]*ka+roll_vel*kav-kt*(Pyset-Py)+kv*Vy);
Vy=Vy+Ay*dt/1000000;
Py=Py+Vy*dt/1000000;

  //saturazione velocità
 if (Vx>vmax)
 Vx=vmax;
  if (Vy>vmax)
 Vy=vmax;
 if (Vx<-vmax)
 Vx=-vmax;
  if (Vy<-vmax)
 Vy=-vmax;
 }

void comando(int data, int dt){
   switch (data) {
 case su:
        Pyset=Pyset+vel*dt/1000000;
   break;
 case giu:
        Pyset=Pyset-vel*dt/1000000;
   break;
 case dx:
        Pxset=Pxset+vel*dt/1000000;
   break;
 case sx:
        Pxset=Pxset-vel*dt/1000000;
   break;
```

```
case su+dx:
      Pxset=Pxset+vel*dt/1000000;
      Pyset=Pyset+vel*dt/1000000;
  break;
case su+sx:
      Pxset=Pxset-vel*dt/1000000;
      Pyset=Pyset+vel*dt/1000000;
  break;
case giu+dx:
      Pxset=Pxset+vel*dt/1000000;
      Pyset=Pyset-vel*dt/1000000;
  break;
case giu+sx:
      Pxset=Pxset-vel*dt/1000000;
      Pyset=Pyset-vel*dt/1000000;
  break;
case X:
      Pxset=0;
      Pyset=-1;
  break;
case O:
      Pxset=1;
      Pyset=0;
  break;
case quadrato:
      Pxset=-1;
      Pyset=0;
  break;
case triangolo:
      Pxset=0;
      Pyset=1;
  break;
case R1+L1:
      Pxset=0;
      Pyset=0;
  break;
}
}
```

Script di cinematica accurata

```
clear
close all

N=9;

d=0.01;
R=0.12;
cost=d/R;
r_w=0.05;
fi0=deg2rad(43);
f1=@(t) -10; %velocità function handle ruota 1
f2=@(t) 5;  %velocità function handle ruota 2
f3=@(t) 5;  %velocità function handle ruota 3

ff1=0;
ff2=15; %15;       %sfasamento ruota 2 risp 1
ff3=21; %21;       %sfasamento ruota 3 risp 1

sen1=(1/(2*(R+r_w)))*(sqrt(4*(R+r_w)^2-d^2)*sin(fi0)-d*cos(fi0));
sen2=1/(2*(R+r_w))*(sqrt(4*(R+r_w)^2-d^2)*sin(fi0)+d*cos(fi0));
cos1=1/(2*(R+r_w))*(sqrt(4*(R+r_w)^2-d^2)*cos(fi0)+d*sin(fi0));
cos2=1/(2*(R+r_w))*(sqrt(4*(R+r_w)^2-d^2)*cos(fi0)-d*sin(fi0));

J0=[ R*cos(fi0) 0 -R*sin(fi0);-R*0.5*cos(fi0) R*sqrt(3/4)*cos(fi0)  -R*sin(fi0);-R*0.5*cos(fi0) -R*sqrt(3/4)*cos(fi0)  -
R*sin(fi0) ];


t=(0:0.001:20)';
xx=0;
yy=0;
zz=0;
V=zeros(3,length(t));
pos=zeros(2,length(t));
caso=zeros(1,length(t));
TT=zeros(3,length(t));

V0=V;
pos0=pos;
theta=[0 0 0];

pos_global=zeros(4,length(t));
pos0_global=pos_global;

angz(1)=0;
angz0(1)=0;



for i=2:length(t)
  % considero gli angoli delle 3 ruote come variabili tra 0 e 360/N
  % superato tale valore ritorna al valore 0

   if abs(theta(1))>360/(N)
      xx=xx+1;
   end
   if abs(theta(2))>360/(N)
      yy=yy+1;
   end
   if abs(theta(3))>360/(N)
```

```
    zz=zz+1;
  end

  theta1=rad2deg(abs(f1(t(i))*t(i)))-360/(N)*xx;
  theta2=rad2deg(abs(f2(t(i))*t(i)))-360/(N)*yy;
  theta3=rad2deg(abs(f3(t(i))*t(i)))-360/(N)*zz;

  theta=[theta1,theta2,theta3];

  TT([1 2 3],i)=theta;
% 8 possibili casi di matrici jacobiane

  if 0<=abs(theta(1))&&abs(theta(1))<360/(2*N) && (0<=abs(theta(2))&&abs(theta(2))<360/(2*N)-ff2 || 360/(N)-
ff2<=abs(theta(2))&&abs(theta(2))<360/(N)) && (0<=abs(theta(3))&&abs(theta(3))<360/(2*N)-ff3 || 360/(N)-
ff3<=abs(theta(3))&&abs(theta(3))<360/(N)) %1
J=[R*cos1 0 -R*sen1;-R*0.5*cos1 R*sqrt(3/4)*cos1  -R*sen1;-R*0.5*cos1 -R*sqrt(3/4)*cos1  -R*sen1 ];
disp('caso 1');
caso(i)=1;
  else if 0<=abs(theta(1))&&abs(theta(1))<360/(2*N) && (0<=abs(theta(2))&&abs(theta(2))<360/(2*N)-ff2 ||
360/(N)-ff2<=abs(theta(2))&&abs(theta(2))<360/(N)) && 360/(2*N)-ff3<=abs(theta(3))&&abs(theta(3))<360/(N)-ff3
%2
J=[R*cos1 0  -R*sen1;-R*0.5*cos1 R*sqrt(3/4)*cos1  -R*sen1;-R*0.5*cos2 -R*sqrt(3/4)*cos2  -R*sen2 ];
disp('caso 2');
caso(i)=2;
      else if 0<=abs(theta(1))&&abs(theta(1))<360/(2*N) && 360/(2*N)-ff2<=abs(theta(2))&&abs(theta(2))<360/(N)-
ff2 && (0<=abs(theta(3))&&abs(theta(3))<360/(2*N)-ff3 || 360/(N)-ff3<=abs(theta(3))&&abs(theta(3))<360/(N)) %3
J=[R*cos1 0  -R*sen1;-R*0.5*cos2 R*sqrt(3/4)*cos2  -R*sen2;-R*0.5*cos1 -R*sqrt(3/4)*cos1  -R*sen1 ];
disp('caso 3');
caso(i)=3;
         else if 0<=abs(theta(1))&&abs(theta(1))<360/(2*N) && 360/(2*N)-
ff2<=abs(theta(2))&&abs(theta(2))<360/(N)-ff2 && 360/(2*N)-ff3<=abs(theta(3))&&abs(theta(3))<360/(N)-ff3 %4
J=[R*cos1 0  -R*sen1;-R*0.5*cos2 R*sqrt(3/4)*cos2  -R*sen2;-R*0.5*cos2 -R*sqrt(3/4)*cos2  -R*sen2 ];
disp('caso 4');
caso(i)=4;
           else if 360/(2*N)<=abs(theta(1))&&abs(theta(1))<360/(N) &&
(0<=abs(theta(2))&&abs(theta(2))<360/(2*N)-ff2 || 360/(N)-ff2<=abs(theta(2))&&abs(theta(2))<360/(N)) &&
(0<=abs(theta(3))&&abs(theta(3))<360/(2*N)-ff3 || 360/(N)-ff3<=abs(theta(3))&&abs(theta(3))<360/(N)) %5
J=[R*cos2 0  -R*sen2;-R*0.5*cos1 R*sqrt(3/4)*cos1 -R*sen1;-R*0.5*cos1 -R*sqrt(3/4)*cos1  -R*sen1 ];
disp('caso 5');
caso(i)=5;
              else if 360/(2*N)<=abs(theta(1))&&abs(theta(1))<360/(N) &&
(0<=abs(theta(2))&&abs(theta(2))<360/(2*N)-ff2 || 360/(N)-ff2<=abs(theta(2))&&abs(theta(2))<360/(N)) &&
360/(2*N)-ff3<=abs(theta(3))&&abs(theta(3))<360/(N)-ff3 %6
J=[R*cos2 0 -R*sen2;-R*0.5*cos1 R*sqrt(3/4)*cos1  -R*sen1;-R*0.5*cos2 -R*sqrt(3/4)*cos2  -R*sen2 ];
disp('caso 6');
caso(i)=6;
                  else if 360/(2*N)<=abs(theta(1))&&abs(theta(1))<360/(N) && 360/(2*N)-
ff2<=abs(theta(2))&&abs(theta(2))<360/(N)-ff2 && (0<=abs(theta(3))&&abs(theta(3))<360/(2*N)-ff3 || 360/(N)-
ff3<=abs(theta(3))&&abs(theta(3))<360/(N)) %7
J=[R*cos2 0  -R*sen2;-R*0.5*cos2 R*sqrt(3/4)*cos2  -R*sen2;-R*0.5*cos1 -R*sqrt(3/4)*cos1  -R*sen1 ];
disp('caso 7');
caso(i)=7;
```

```
                    else if 360/(2*N)<=abs(theta(1))&&abs(theta(1))<360/(N) && 360/(2*N)-
ff2<=abs(theta(2))&&abs(theta(2))<360/(N)-ff2 && 360/(2*N)-ff3<=abs(theta(3))&&abs(theta(3))<360/(N)-ff3  %8
J=[R*cos2 0  -R*sen2;-R*0.5*cos2 R*sqrt(3/4)*cos2  -R*sen2;-R*0.5*cos2 -R*sqrt(3/4)*cos2  -R*sen2 ];
disp('caso 8');
caso(i)=8;
                        end
                    end
                end
            end
        end
    end
end

%% cinematica accurata
w_w=[f1(t(i));f2(t(i));f3(t(i))];  %velocità angolare delle ruote dalla function handle
v_wheel=w_w.*r_w;                  %velocità tangenziale del punto di contatto ruota-sfera
W(:,i)=inv(J)*v_wheel;             %velocità angolare della sfera nel sdr locale ballbot
V([2 1],i)=W([1 2],i)*R;
V(2,i)=-V(2,i);
pos(:,i)=pos(:,i-1)+V([1 2],i).*(t(i)-t(i-1));

%% cinematica approssimata
W0(:,i)=inv(J0)*v_wheel;
V0([2 1],i)=W0([1 2],i)*R;
V0(2,i)=-V0(2,i);
pos0(:,i)=pos0(:,i-1)+V0([1 2],i).*(t(i)-t(i-1));
% pos0(:,i)=V0([1 2],i).*(t(i));


% theta
% [xx yy zz]
% inv(J)
% V(:,i)
ang_error(i)=rad2deg(acos((dot(W0(:,i),W(:,i)))/(norm(W0(:,i))*norm(W(:,i)))));


%% posizione globale accurata
angz(i)=angz(i-1)+W(3,i)*(t(i)-t(i-1));
% angz(i)=W(3,i)*(t(i)-t(i-1));
pos_global([1 2],1)=pos(:,2)-pos(:,1);
dp=V([1 2],i).*(t(i)-t(i-1));

Ai0=[cos(angz(i)) -sin(angz(i)) 0 pos_global(1,i-1);...
    sin(angz(i))  cos(angz(i)) 0 pos_global(2,i-1);...
     0       0     1 0;...
     0       0     0  1];

Aus=Ai0*[1 0 0 dp(1);0 1 0 dp(2);0 0 1 0; 0 0 0 1];
pos_global(:,i)= Aus(:,4);

%% posizione globale approssimata
angz0(i)=angz0(i-1)+W0(3,i)*(t(i)-t(i-1));
% angz0(i)=W0(3,i)*(t(i)-t(i-1));
pos0_global([1 2],1)=pos0(:,2)-pos0(:,1);
dp0=V0([1 2],i).*(t(i)-t(i-1));
```

```
Ai0_0=[cos(angz0(i)) -sin(angz0(i)) 0 pos0_global(1,i-1);...
    sin(angz0(i))  cos(angz0(i)) 0 pos0_global(2,i-1);...
     0       0     1 0;...
     0       0     0  1];

pos0_global(:,i)=Ai0_0*[dp0; 0; 1];

end


ang_error(1)=ang_error(2);



for i=1:length(t)
  norm_w(i)=norm(W(:,i));
  norm_w0(i)=norm(W0(:,i));
  norm_pos(i)=norm(pos_global([1 2],i)-pos0_global([1 2],i));
end


for j=1:length(t)
   if caso(j)==0
      caso(j)=caso(j+1);
   end
end
%%
figure(3)
plot(pos(1,:),pos(2,:),'-r',pos0(1,:),pos0(2,:),'-b','LineWidth',1); grid on;
xlabel('Asse x [m]');
ylabel('Asse y [m]');
legend('posizione accurata','posizione approssimata');
title('Posizione nello spazio nel sdr locale del robot')

figure(11)
subplot(1,2,2)
plot(t,ang_error,'-r','LineWidth',1); grid on;
ylabel('Errore angolare [°]');
xlabel('tempo [s]');
title('errore angolare del vettore velocita'')


subplot(1,2,1)
plot(t,abs(norm_w-norm_w0)./norm_w*100,'-b','LineWidth',1); grid on;
ylabel('Errore modulo velocità [%]');
xlabel('tempo [s]');
title('errore in modulo del vettore velocita'')

figure(10)
subplot(2,1,1)
plot(t,ang_error,'-r','LineWidth',1); grid on;
ylabel('Errore angolare [°]');
xlabel('tempo [s]');
title('errore angolare del vettore velocita'')

subplot(2,1,2)
plot(t,caso,'-b','LineWidth',1);grid on;
xlabel('tempo [t]');
ylabel('Caso');
title('Caso specifico di matrice jacobiana');

 figure(5)
plot(pos_global(1,:),pos_global(2,:),'-r',pos0_global(1,:),pos0_global(2,:),'-b'); grid on;
```

```
xlabel('Asse x [m]');
ylabel('Asse y [m]');
legend('posizione globale accurata','posizione globale approssimata');
title('Posizione nello spazio nel sdr globale')

figure(6)
subplot(1,2,1)
plot(t(2:end),V(1,2:end),'r',t(2:end),V0(1,2:end),'--r','LineWidth',1); grid on;
legend("accurata","approssimata");
xlabel('tempo [s]');
ylabel('velocita" V_x [m/s]');

subplot(1,2,2)
plot(t(2:end),V(2,2:end),'b',t(2:end),V0(2,2:end),'--b','LineWidth',1); grid on;
legend("accurata","approssimata");
xlabel('tempo [s]');
ylabel('velocita" V_y [m/s]');

 figure(9)
plot(t,caso,'-b','LineWidth',1);grid on;
xlabel('tempo [t]');
ylabel('Caso');
title('Caso specifico di matrice jacobiana');

 TT(1,:)=TT(1,:)+ff1;
TT(2,:)=TT(2,:)+ff2;
TT(3,:)=TT(3,:)+ff3;
figure(1)
plot(t,TT(1,:),'-k',t,TT(2,:),'-b',t,TT(3,:),'-r','LineWidth',1);grid on;
xlabel('tempo [t]');
ylabel('angolo [°]');
title('angoli delle 3 ruote omnidirezionali');

figure(8)
plot(t,norm_pos,'b','LineWidth',1);grid on;
xlabel('tempo [t]');
ylabel('modulo errore sulla posizione  [m]');
title('Errore di posizione nel sdr globale')

figure(17)
subplot(2,1,1)
plot(pos_global(1,:),pos_global(2,:),'-r',pos0_global(1,:),pos0_global(2,:),'-b'); grid on;
xlabel('Asse x [m]');
ylabel('Asse y [m]');
legend('posizione globale accurata','posizione globale approssimata');
title('Posizione nello spazio nel sdr globale')

subplot(2,1,2)
plot(pos(1,:),pos(2,:),'-r',pos0(1,:),pos0(2,:),'-b','LineWidth',1); grid on;
xlabel('Asse x [m]');
ylabel('Asse y [m]');
legend('posizione accurata','posizione approssimata');
title('Posizione nello spazio nel sdr locale del robot')
```

Script di lettura dati sperimentali

```
close;
clear;

% Nome della prova (inclusa estensione)
test_name = 'stabilità_14_3_1_2.xlsx';

% Lettura del file txt
acq = importdata(test_name);
%%
t=linspace(acq.data(1,1),acq.data(end,1),size(acq.data,1));
t=t-acq.data(1,1);

figure(1);
subplot(1,2,1)
plot(t,acq.data(:,2),'-r','linewidth',.2); grid on;
xlabel('tempo [s]');
ylabel('\theta y [°]');
subplot(1,2,2)
plot(t,acq.data(:,3),'-b','linewidth',.2); grid on;
xlabel('tempo [s]');
ylabel('\theta x [°]')

figure(2)
subplot(1,2,1)
plot(t,acq.data(:,4),'-r','linewidth',.2); grid on;
title('velocità angolare')
xlabel('tempo [s]');
ylabel('\theta_y dot [°/s]')
subplot(1,2,2)
plot(t,acq.data(:,5),'-b','linewidth',.2); grid on;
title('velocità angolare ')
xlabel('tempo [s]');
ylabel('\theta_x dot [°/s]')

figure(3)
subplot(1,2,1)
plot(t,acq.data(:,6),'-r','linewidth',.2); grid on;
xlabel('tempo [s]');
ylabel('V_x [m/s]')
subplot(1,2,2)
plot(t,acq.data(:,7),'-b','linewidth',.2); grid on;
xlabel('tempo [s]');
ylabel('V_y [m/s]')

figure(4)
subplot(1,2,1)
plot(t,acq.data(:,8),'-r','linewidth',.2); grid on;
xlabel('tempo [s]');
ylabel('X [m]')
subplot(1,2,2)
plot(t,acq.data(:,9),'-b','linewidth',.2); grid on;
xlabel('tempo [s]');
ylabel('Y [m]')
%%
figure(5)
subplot(3,1,1)
plot(t,acq.data(:,2),'-r','linewidth',.2); grid on;
```

```
xlabel('tempo [s]');
ylabel('\theta y [m]')
subplot(3,1,2)
plot(t,acq.data(:,6),'-b','linewidth',.2); grid on;
xlabel('tempo [s]');
ylabel('V_X [m]')
subplot(3,1,3)
plot(t,acq.data(:,8),'-r','linewidth',.2); grid on;
xlabel('tempo [s]');
ylabel('X [m]')

%%
figure(6)
subplot(2,2,1)
plot(t,acq.data(:,2).*(pi/180),'-r','linewidth',.8); grid on;
xlabel('tempo [s]');
hold on;
plot(t,acq.data(:,6),'-b','linewidth',.8); grid on;
xlabel('tempo [s]');
hold on;
plot(t,acq.data(:,8),'-g','linewidth',.8); grid on;
xlabel('tempo [s]');
subplot(2,2,2)
plot(t,acq.data(:,2).*(pi/180),'-r','linewidth',.8); grid on;
xlabel('tempo [s]');
hold on;
plot(t,acq.data(:,6),'-b','linewidth',.8); grid on;
xlabel('tempo [s]');
hold on;
plot(t,acq.data(:,8),'-g','linewidth',.8); grid on;
xlabel('tempo [s]');
subplot(2,2,3)
plot(t,acq.data(:,2).*(pi/180),'-r','linewidth',.8); grid on;
xlabel('tempo [s]');
hold on;
plot(t,acq.data(:,6),'-b','linewidth',.8); grid on;
xlabel('tempo [s]');
hold on;
plot(t,acq.data(:,8),'-g','linewidth',.8); grid on;
xlabel('tempo [s]');
subplot(2,2,4)
plot(t,acq.data(:,2).*(pi/180),'-r','linewidth',.8); grid on;
xlabel('tempo [s]');
hold on;
plot(t,acq.data(:,6),'-b','linewidth',.8); grid on;
xlabel('tempo [s]');
hold on;
plot(t,acq.data(:,8),'-g','linewidth',.8); grid on;
xlabel('tempo [s]');
```

BIBLIOGRAFIA

[1] T.B.Lauwers, G.A.Kantor, R.L.Hollis, "A Dynamically Stable Single-Wheeled Mobile Robot with Inverse Mouse-Ball Drive" Proc. ICRA2006, pp 2884–2889

[2] Kumagai M, Ochiai T. "Development of a robot balancing on a ball". International conference on control, automation and systems. Seoul, Korea;2008.

[3] T.B. Lauwers, G. Kantor, and R.L. Hollis. "One is enough!" In Proc. Int'l. Symp. for Robotics Research, October 12-15 2005.

[4] A. Weiss, R.G. Langlois, M.J.D. Hayes. "The effects of dual row omnidirectional wheels on the kinematics of the Atlas spherical motion platform" Mechanism and Machine Theory 44 (2009) 349–358

[5] https://www.arduino.cc

[6] Fankhauser, Péter; Gwerder, Corsin, "Modeling and Control of a Ballbot", 2010, Eidgenössische Technische Hochschule Zürich, DOI:10.3929/ethz-a-010056685