POLITECNICO DI TORINO

CORSO DI LAUREA MAGISTRALE IN

INGEGNERIA MECCANICA - AUTOMAZIONE

TESI DI LAUREA MAGISTRALE

Progettazione e sviluppo dell'architettura di controllo di un dispositivo di assistenza alla propulsione per sedie a rotelle

Relatore:
Prof. Pastorelli Stefano Paolo
Correlatore:
Ing. Cornagliotto Valerio
Ing. Polito Michele

Candidato:
Falcone Donato

ANNO ACCADEMICO 2022-23
OTTOBRE 2023

Abstract

L'uso di sedie a rotelle manuali è molto diffuso nella società, ma ci sono diversi fattori che possono renderne l'uso difficoltoso e inefficiente. Per questo motivo esistono in commercio diversi dispositivi di assistenza elettrica per le sedie a rotelle manuali, che si differenziano per il design, per la posizione e per il grado di assistenza.
La tesi affronta la progettazione per una migliore ingegnerizzazione di un dispositivo di assistenza alla propulsione, in grado di assistere l'utente sia in spinta sia in fase di manovra, interpretando le azioni esercitate sugli anelli di spinta. Tale dispositivo presentava delle criticità strutturali e un'architettura hardware e software poco robusta. In una prima fase è stato riprogettato il telaio, per irrigidire la struttura e rendere il dispositivo più compatto. Inoltre, sono state disegnati dei carter di copertura funzionali ed estetici.
In seguito, è stato riprogettato il sistema meccatronico sostituendo alcune componenti elettroniche con l'obiettivo di ottimizzare l'architettura hardware. Le nuove componenti elettroniche sono state testate per verificare la loro piena integrazione nel sistema. In particolare sono state eseguite prove di caratterizzazione del motore di spinta e prove di frenata rigenerativa.
Si è quindi sviluppato il comparto software con l'implementazione delle logiche di controllo sia per la coppia di spinta, sia per la sterzata della ruota del dispositivo. Per permettere prestazioni elevate di controllo e comunicazione, l'architettura software è stata sviluppata in ambiente ROS 2.
Infine sono stati effettuati test a banco per verificare il funzionamento del nuovo sistema.

Indice

Elenco delle figure

Elenco delle tabelle

ACRONIMI

WHO World Health Organization

MWC Manual WheelChair

PAD Power-Assisted Devices

PAPAW Pushrim Activated Power-assisted Wheelchair

SBC Single-Board Computer

GPIO General-Purpose Input/Output

DC Direct Current

BLDC Brushless Direct Current

PMSM Permanent Magnet Synchronous Motor

ACIM Alternative Current Induction Motor

FOC Field-Oriented Control

IMU Inertial Measurement Unit

ROS Robot Operating System

LTS Long Term Support

colcon Cmake Ordered Language CONsolidation

BLE Bluetooth Low Energy

Capitolo 1

INTRODUZIONE

La sedia a rotelle è uno dei dispositivi di assistenza più usati per migliorare la mobilità. Da studi effettuati nel 2010 da parte dell'WHO risulta che circa il 10% della popolazione mondiale ha una disabilità, circa 650 milioni di persone; di queste circa il 10% necessità dell'uso di una sedia a rotelle [1]. Stime più recenti, invece, indicano che circa 131,8 milioni di persone nel mondo necessitano l'uso di una sedia a rotelle, che corrisponde a circa l'1,85% della popolazione mondiale [2].
Le categorie di persone che necessitano maggiormente di una sedia a rotelle sono [3]:

- **Bambini:** essi necessitano di accesso alla sedia a rotelle non appena viene indicata una disabilità motoria, al fine di massimizzare lo sviluppo fisico, cognitivo e sociale.
- **Anziani:** il numero di persone con età superiore ai 65 anni sta crescendo in modo esponenziale; questa categoria di persone presenta un alto tasso di difficoltà motorie dovuto alla maggiore fragilità.
- **Persone con disabilità:** molte persone con disabilità necessitano dell'uso di carrozzine. Si tratta di persone con patologie diverse come amputazioni, paralisi cerebrale, sclerosi multipla, distrofia muscolare o lesioni del midollo spinale.
- **Condizioni patologiche croniche:** categoria in costante crescita a livello globale a causa dell'aumento di malattie cardiovascolari, diabete e ictus.

Le sedie a rotelle manuali possono promuovere uno stile di vita fisicamente attivo e sono una soluzione più maneggevole e trasportabile rispetto alle sedie a rotelle elettriche [4]. Tuttavia, l'uso di sedie a rotelle manuali (Manual WheelChair (MWC)), può provocare difficoltà agli utenti nel muoversi e nello svolgere le normali attività quotidiane. Le difficoltà potrebbero essere dovute a una ridotta capacità fisica, alla debolezza della parte superiore del corpo, a dolori, a lesioni o all'affaticamento dovuto alla propulsione della sedia a rotelle per un periodo di tempo prolungato [5]. Inoltre, è stato dimostrato che l'efficienza di propulsione delle sedie a rotelle manuali è compresa tra il 5% e il 18% [6–8]. Questa bassa efficienza ne rende difficile o impossibile l'uso efficace da parte di alcune persone [9]. C'è poi da considerare che la cinematica di spinta non è adatta all'anatomia degli arti superiori e quindi crea lesioni o dolore a spalle, gomiti e polsi [10]. Infatti, è stato riportato che l'incidenza di lesioni al polso, gomito e spalla è compresa tra il 25% e l'80% per gli utenti di sedie a rotelle manuali [9].
Fino al 2000 esistevano essenzialmente tre alternative per le persone che non erano in grado di spingere efficacemente una MWC: una sedia a rotelle elettrica, uno scooter elettrico o la spinta da parte di un assistente [9]. Per ovviare ai diversi problemi descritti in precedenza che l'utilizzo delle MWC comporta agli utenti, negli ultimi 20 anni, sono stati sviluppati alcuni dispositivi di assistenza elettrica (Power-Assisted Devices (PAD)) alla propulsione che si possono montare e

smontare da una MWC; questi dispositivi vengono comunemente chiamati add-on. Alcuni studi hanno dimostrato che i dispositivi di assistenza elettrica hanno influenzato in maniera positiva la capacità di propulsione, riducendo lo sforzo fisico legato alla spinta manuale e migliorando la mobilità durante le attività quotidiane [5]-[10].
I dispositivi di assistenza elettrica alla propulsione attualmente in commercio si possono dividere in 4 categorie [11]-[12]:

- **Comando joystick** *(Fig. 1.1a)***:** questi dispositivi forniscono un'assistenza completa alla propulsione comandando i motori tramite joystick. I motori possono essere integrati sul mozzo ruota, possono essere in contatto con le ruote motrici applicando una trasmissione a frizione, oppure l'unità motrice può essere una ruota supplementare montata nella parte inferiore o posteriore del telaio della MWC. Solitamente il joystick viene montato sul bracciolo della MWC. I dispositivi comandati con joystick eliminano completamente la componente attiva dell'utente nella propulsione, questi tipi di dispositivo trovano ambia applicazione in ambienti esterni. La latenza del comando joystick, e la scarsa intuitività di utilizzo del comando rende poco agevole e spesso inappropriato l'utilizzo indoor. Quando i motori sono disinseriti la carrozzina può funzionare come una sedia a rotelle manuale, tuttavia il peso aggiunto dai motori, dovuto al pacco batterie, all'elettronica e al joystick (circa 20kg) rende molto faticoso l'utilizzo della carrozzina in modo manuale.
- **Motoruota anteriore** *(Fig. 1.1b)***:** sono dispositivi costituiti da una ruota motrice collegata ad uno sterzo, fissati alla parte anteriore della sedia a rotelle. Questi dispositivi trasformano la MWC in un triciclo motorizzato, sollevandone la parte anteriore. Il manubrio è dotato di acceleratore e freno, che permettono il completo controllo della MWC. L'aggiunta di una motoruota anteriore rende il funzionamento della carrozzina simile a quello di uno scooter elettrico e quando il dispositivo è collegato la propulsione manuale risulta impossibile, a meno che la motoruota non venga rimossa. Anche questo tipo di dispositivi, come i precedenti, elimina completamente la componente attiva dell'utente nella propulsione. L'aggiunta della motoruota nella parte anteriore della MWC comporta un aumento della lunghezza complessiva della carrozzina e quindi un raggio di sterzata maggiore che li rende inadatti all'utilizzo indoor. In ambienti aperti, invece, questa problematica non si presenta e il dispositivo consente in generale spostamenti più rapidi.
- **Moltiplicatori di spinta** (Pushrim Activated Power-assisted Wheelchair (PAPAW)) *(Fig. 1.1c)***:** sono ruote motorizzate che sostituiscono le ruote originali e forniscono una coppia motrice proporzionale alla spinta esercitata dall'utente sui pushrim. Questi dispositivi diminuiscono la forza necessaria alla propulsione e nel contempo consentono il movimento della sedia a rotelle tra una spinta e un'altra. Inoltre, l'utente ha la possibilità di regolare il grado di assistenza in base alle condizioni di utilizzo. Le ruote motorizzate aggiungono un peso importante alla sedia a rotelle e ne aumenta anche la larghezza. Anche in questo caso la latenza di comando e la scarsa sensibilità degli anelli di spinta può generare coppie di assistenza eccessive della carrozzina, generando comportamenti indesiderati e poco adatti ad un ambiente indoor.
- **Motoruota posteriore** *(Fig. 1.1d)***:** consistono in una ruota motorizzata aggiuntiva montata nella parte posteriore della MWC che fornisce una spinta in direzione longitudinale. Questo tipo di dispositivi viene generalmente controllato tramite braccialetto o manopola, da cui si può impostare la velocità desiderata. La sterzata, invece, viene completamente gestita fisicamente dall'utente che, esercitando una coppia frenante sui pushrim, determina la direzione della MWC. Le motoruote posteriori sono dispositivi facilmente montabili e smontabili, leggeri e di piccole dimensioni, senza comportare un aumento della lunghezza complessiva della sedia a rotelle. Tuttavia, il controllo attualmente implementato consiste

nell'impostare un valore di velocità costante. Questo tipo di controllo rende completamente inadatto il dispositivo in ambienti indoor ristretti. Inoltre, per gestire la direzione della MWC l'utente deve frenare, esercitando un'azione negativa da un punto di vista energetico. Questo metodo di controllo della direzione risulta poco efficiente, faticoso e a volte doloroso per l'utente.

(a) *Comando Joystick [13]*

(b) *Motoruota anteriore [14]*

(c) *Moltiplicatori di spinta [15]*

(d) *Motoruota posteriore [16]*

Figura 1.1: Categorie di dispositivi di assistenza

Nelle prime due categorie si parla di assistenza completa perché all'utente non è richiesto nessun contributo attivo per spingere la carrozzina, aspetto negativo da un punto di visto psico-fisico. Nel caso di PAPAW e motoruota posteriore, invece, vi è assistenza parziale poiché per i primi è necessario un comando fisico per ogni azione e per i secondi bisogna esercitare una coppia nel momento in cui si vuole sterzare, anche se in questo caso il contributo attivo richiesto all'utente risulta eccessivamente faticoso, energeticamente svantaggioso e poco intuitivo.
I dispositivi montati anteriormente sono quelli che risultano più semplici da montare, ma che presentano l'ingombro maggiore rispetto agli altri. I PAPAW e le motoruote posteriori sono le soluzioni che incidono meno sull'ingombro complessivo della carrozzina. Inoltre, le motoruote

posteriori sono anche i dispositivi più leggeri, e ciò fa sì che siano anche i più facilmente trasportabili rispetto alle altre soluzioni. Si possono poi analizzare anche differenze per quanto riguarda il funzionamento e il controllo dei dispositivi. I PAPAW rilevano l'intenzione dell'utente attraverso la misura della forza esercitata sui pushrim e ogni ruota eroga una coppia aggiuntiva proporzionale, permettendo così all'utente di percepire uno sforzo minore. Inoltre, garantiscono assistenza anche in frenata dal momento che quando l'utente esercita una forza frenante sui pushrim, il dispositivo applica una forza proporzionale contraria alla rotazione delle ruote. Le motoruote anteriori e i dispositivi comandati con joystick sono completamente controllati rispettivamente dal manubrio e dal joystick. Sulle motoruote posteriori è implementato un controllo della velocità, gestito tramite braccialetto o manopola. Questo tipo di dispositivi, però, non presenta un'assistenza attiva alla frenata, per cui l'utente deve esercitare fisicamente tutta la coppia frenante necessaria a fermare la MWC.

Sulla base delle considerazioni fatte sui pro e contro dei dispositivi di assistenza alla propulsione precedentemente descritti, è stato recentemente realizzato il prototipo di un dispositivo di assistenza alla propulsione che integri in un solo dispositivo il comando intuitivo proprio dei dispositivi PAPAW e la compattezza e la leggerezza dei dispositivi motoruota posteriori. Questo prototipo si differenzia dai dispositivi commerciali per i seguenti punti: il comando del dispositivo avviene tramite l'informazione di forza registrata sui pushrim e sul dispositivo è implementato un tipo di controllo adattativo, che tiene in considerazione sia le spinte effettuate dall'utente, sia la dinamica della carrozzina. L'utilizzo dei pushrim come dispositivo di comando consente comunque all'utente di svolgere dell'attività fisica con la parte superiore del corpo, beneficiandone sia da un punto di vista metabolico, che da un punto di vista psicologico. Inoltre, è stata aggiunta la funzionalità di una ruota sterzante controllata attivamente per assistere l'utente anche durante fasi di manovra e curve.
Il mio lavoro di tesi è stato quello di apportare alcune modifiche all'architettura del dispositivo, riprogettando alcuni aspetti e migliorando l'ingegnerizzazione di alcune componenti meccaniche. Il primo prototipo presentava alcune problematiche sull'architettura complessiva per cui sono state implementate alcune modifiche:

- progettazione di alcuni componenti della **struttura meccanica** e di design del dispositivo;
- sostituzione del **driver del motore**, per poter implementare:
  - **controllo coppia**, invece di un controllo velocità;
  - assistenza di **frenata rigenerativa**;
- sostituzione del **microprocessore** e utilizzo di un sistema operativo specifico per applicazioni robotiche (**ROS 2**);
- implementazione di un'**architettura software multisensore** che permetta il funzionamento in parallelo dei componenti.

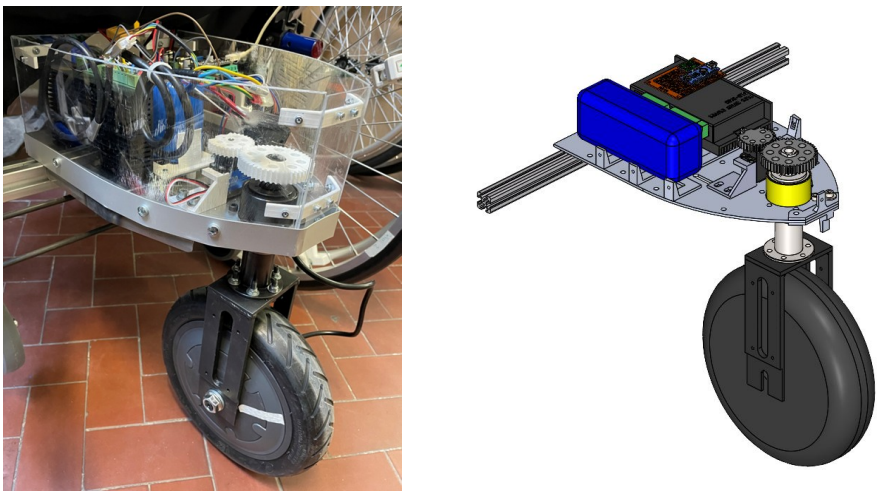
Capitolo 2

STRUTTURA

In questo capitolo si analizzerà la struttura meccanica del dispositivo, partendo dalla descrizione della soluzione costruttiva del primo prototipo, per evidenziarne le criticità. Sono state quindi introdotte nuove soluzioni progettuali, riportando la modellazione e produzione di alcuni componenti. A supporto delle modifiche effettuate sono state eseguite delle simulazioni strutturali ad elementi finiti (FEM). Infine, è descritto il design della copertura ideata per la nuova soluzione costruttiva.

2.1 Struttura primo prototipo

In figura 2.1 viene riportata la struttura complessiva del primo prototipo. In particolare, si osserva che il dispositivo è composto da un profilato orizzontale che funge da collegamento con la carrozzina. Alle estremità di tale profilato sono state realizzate delle interfacce che ne permettono l'accoppiamento all'asse delle ruote della MWC. Sullo stesso profilato è stata montata una piastra di alluminio sagomata e forata che costituiva la struttura principale del dispositivo. In figura 2.1 è possibile osservare che nella parte anteriore della piastra erano fissate la batteria, il driver del motore e le altre schede di controllo. Nella parte centrale, era montato il servomotore la cui funzione era quella di impostare l'angolo di sterzata della ruota, mentre un accoppiamento tramite ruote dentate permetteva la rotazione della ruota stessa.



(a) *Foto del prototipo* **(b)** *Rappresentazione CAD*

Figura 2.1: Primo prototipo

In figura 2.2 si può osservare una vista in sezione della parte posteriore del dispositivo, dalla quale si può notare meglio l'accoppiamento meccanico tra le ruote dentate. In particolare, la ruota dentata con diametro maggiore è calettata su un albero cavo. Quest'ultimo è accoppiato nella parte superiore con l'anello interno di un cuscinetto a doppia corona di sfere. Si nota che il cuscinetto è inserito all'interno di una sede (parte gialla in fig. 2.2) che, oltre ad accoppiarsi radialmente al cuscinetto, ne blocca la parte bassa dell'anello esterno. Nella parte superiore l'anello esterno è bloccato da un coperchio avvitato sulla sede con un risalto che va proprio a contatto con il cuscinetto, mentre l'anello interno è bloccato superiormente da un anello seeger e nella parte inferiore dallo spallamento dell'albero. La sede del cuscinetto è inserita in un foro creato nella piastra e tramite una parte flangiata vi viene montata. Inoltre, dalla figura 2.2 si può osservare che la parte inferiore dell'albero è filettata e va ad accoppiarsi con un secondo albero cavo che presenta una filettatura interna. Avere i due alberi filettati serve per regolare l'angolo degli stessi rispetto al terreno; per bloccare la posizione relativa tra i due è interposta una ghiera. È possibile notare che il secondo albero presenta una flangia sull'estremità inferiore che ne permette il collegamento con la forcella.

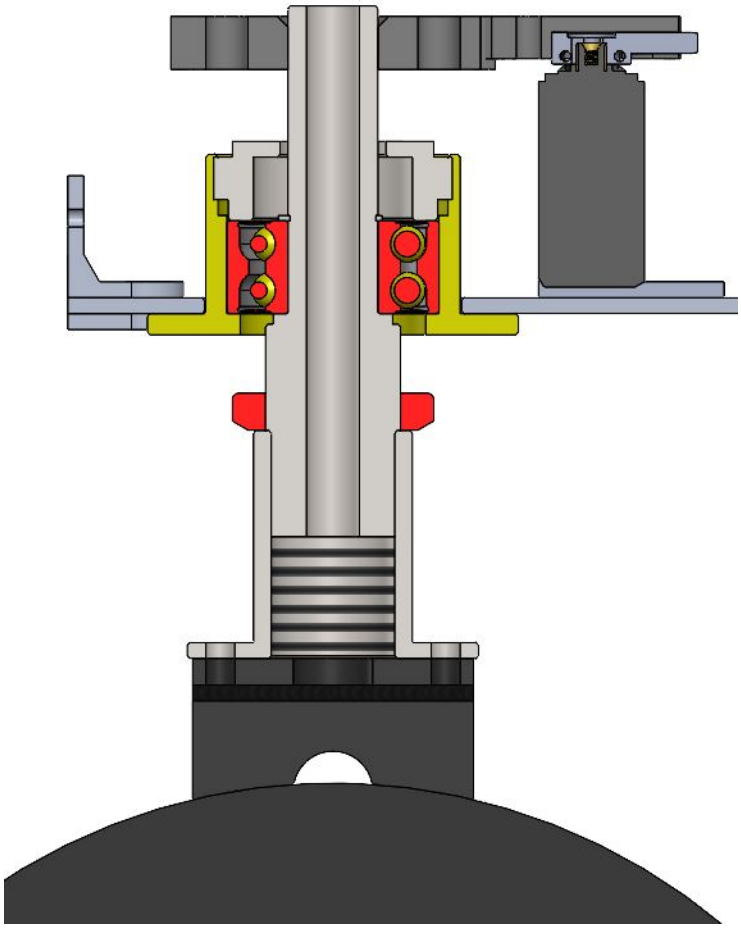


Figura 2.2: Sezione albero di sterzo del primo prototipo

Nella parte inferiore della forcella è montata una ruota motorizzata di tipo brushless; nello specifico si tratta di una ruota per e-scooter, di diametro pari a 230 mm. Nella figura 2.3 è possibile osservare il sistema complessivo montato ad una MWC.

Questo prototipo presentava alcune criticità dal punto di vista meccanico: in particolare la struttura era poco rigida e la disposizione degli elementi all'interno del dispositivo non era ottimale. Infatti, durante le fasi di spinta e sterzata la piastra subiva importanti flessioni che potevano portare alla rottura del dispositivo stesso. Inoltre lo smontaggio e il montaggio del dispositivo risultava scomodo in quanto prevedeva la rimozione di una delle ruote per poter scollegare il dispositivo dalla carrozzina.

Figura 2.3: Primo prototipo in utilizzo

2.2 Nuove soluzioni costruttive

Nella progettazione di una nuova soluzione di dispositivo si è prima focalizzata l'attenzione sulla risoluzione del problema relativo alla poca rigidità della struttura e per questo sono stati utilizzati dei profilati modulari, che offrono un buon rapporto tra peso e proprietà meccaniche. Un esempio dei profilati utilizzati è riportato in figura 2.4 e le loro caratteristiche principali sono riportate in tabella 2.1

Figura 2.4: Profilato utilizzato

Proprietà	
Materiale	Lega di alluminio
Designazione materiale	EN AW – 6060
Stato	Indurito a caldo
Finitura	Anodizzato naturale
Sezione trasversale	20x20 mm
Peso per unità di lunghezza	0.48 kg/m
Resistenza a trazione R_m	245 N/mm^2
Durezza	75 HB
Momento d'inerzia a torsione I_t	0.07 cm^4
Limite di snervamento $R_{p0,2}$	195 N/mm^2
Momento d'inerzia trasversale I_x, I_y	0.72 cm^4
Momento resistente trasversale W_x, W_y	0.72 cm^3

Tabella 2.1: Dati tecnici dei profilati utilizzati

L'idea alla base delle nuove soluzioni strutturali che saranno presentate in seguito è quella di sfruttare maggiormente lo spazio in direzione verticale per far sì che l'aggiunta del dispositivo alla MWC non vada a modificare nel complesso la lunghezza della stessa. Inoltre, i profilati utilizzati permettono una notevole modularità e la possibilità di vagliare diverse soluzioni costruttive, utilizzando il materiale solo dove realmente necessario.
Nella figura 2.5 è presente uno schema esemplificativo della struttura, in cui non sono riportati i profilati modulari descritti prima; questo perché l'intento è solo quello di mostrare l'idea del prodotto finale da ottenere. Questa soluzione preliminare, infatti, è risultata eccessivamente sovradimensionata rispetto alle sollecitazioni in gioco, oltre che risultare eccessiva anche in termini di peso.

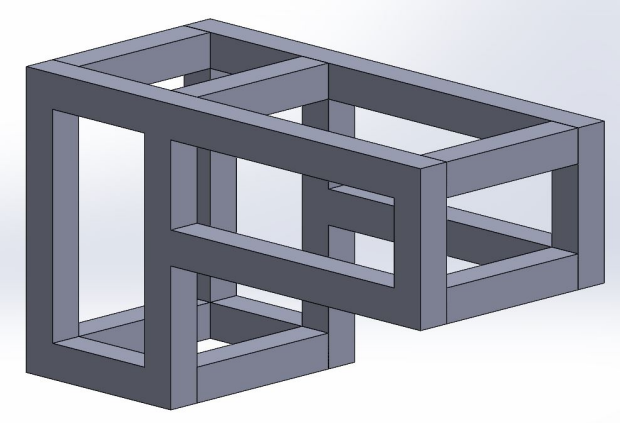


Figura 2.5: Prima soluzione strutturale

Per i motivi sopra descritti è stata modificata la struttura inserendo solo i profilati e i supporti angolari necessari a garantire la rigidità e la resistenza richiesta (Fig. 2.6). Dalla figura è possibile notare che nella parte posteriore i due profilati laterali non sono collegati tra di loro. Infatti, si è resa necessaria la progettazione di un'interfaccia con tolleranze più stringenti per permettere l'accoppiamento con i componenti di supporto all'albero di sterzo, la stessa funge anche da collegamento tra i due profilati laterali. Per queste due ragioni, quindi, si è deciso deciso di realizzare una piccola piastra in acciaio che fungesse da elemento di collegamento sia tra i due profilati laterali della struttura sia per il blocco cuscinetto. Si riporta una rappresentazione in figura 2.7 e nell'appendice A.1 si allega la tavola eseguita per la realizzazione del componente.

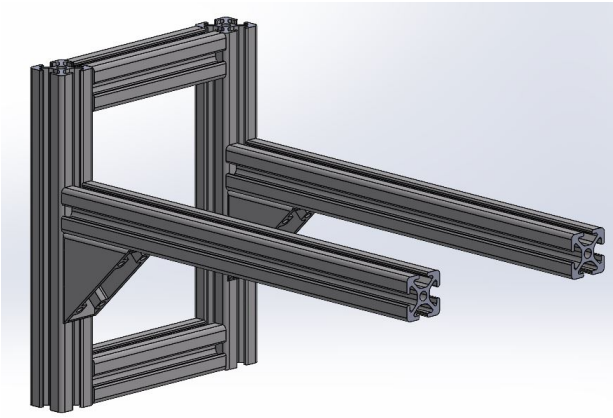


Figura 2.6: Seconda soluzione strutturale

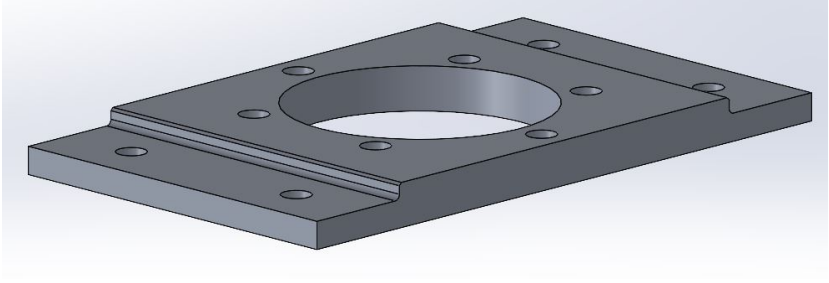


Figura 2.7: Piastra di collegamento tra il telaio e i componenti di sterzo

La struttura meccanica finale rappresentata in figura 2.8 è molto simile alla precedente, con la differenza che nella zona anteriore è stata cambiata la posizione del profilato orizzontale, inserendolo nella parte centrale, con lo scopo di darvi maggiore rigidità. In più, nella parte superiore sono stati aggiunti due profili angolari simmetrici, uno nella zona anteriore e uno in quella posteriore, e in corrispondenza di quest'ultima sono stati inseriti anche due profilati verticali necessari al collegamento del secondo profilo angolare. L'aggiunta dei due componenti dà la possibilità di inserire un elemento di presa efficace nella zona del baricentro della struttura.

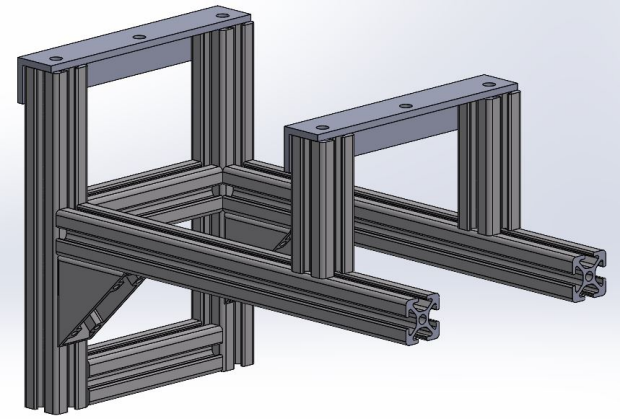


Figura 2.8: Soluzione strutturale definitiva

Inoltre, rispetto alla soluzione precedente, i due alberi coassiali sono stati sostituiti da un unico albero (Fig. 2.9) che si interfaccia con l'anello interno del cuscinetto e su cui è stata saldata la flangia di collegamento con la forcella. Anche per questo componente nell'appendice A.2 è allegata la tavola realizzata.

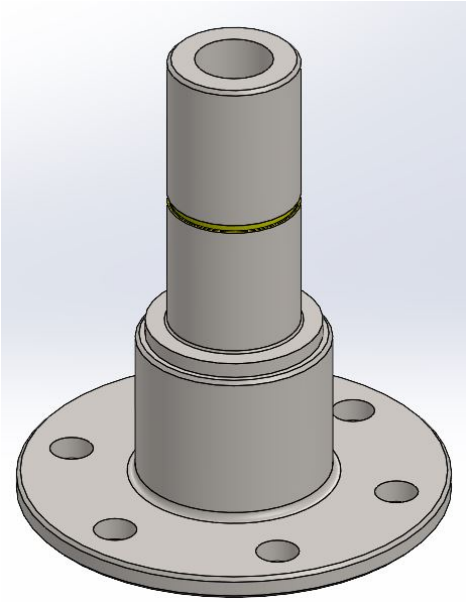


Figura 2.9: Albero di collegamento tra forcella e motore di sterzo

Della struttura precedente sono state utilizzate la forcella, il blocco cuscinetto e il profilato di collegamento con la carrozzina.

La struttura meccanica definitiva è rappresentata nella figura 2.10, dove si osserva che la struttura si collega al profilato di collegamento alla carrozzina tramite degli angolari che permettono il collegamento a diverse altezze rispetto ai profili verticali. Ciò consente di variare l'angolo tra il terreno e l'asse dell'albero e in questo modo, quindi, si può testare la risposta della struttura a diverse angolazioni della forza di spinta. Nello specifico in questo prototipo è possibile cambiare l'incidenza di $\pm$ 10°.

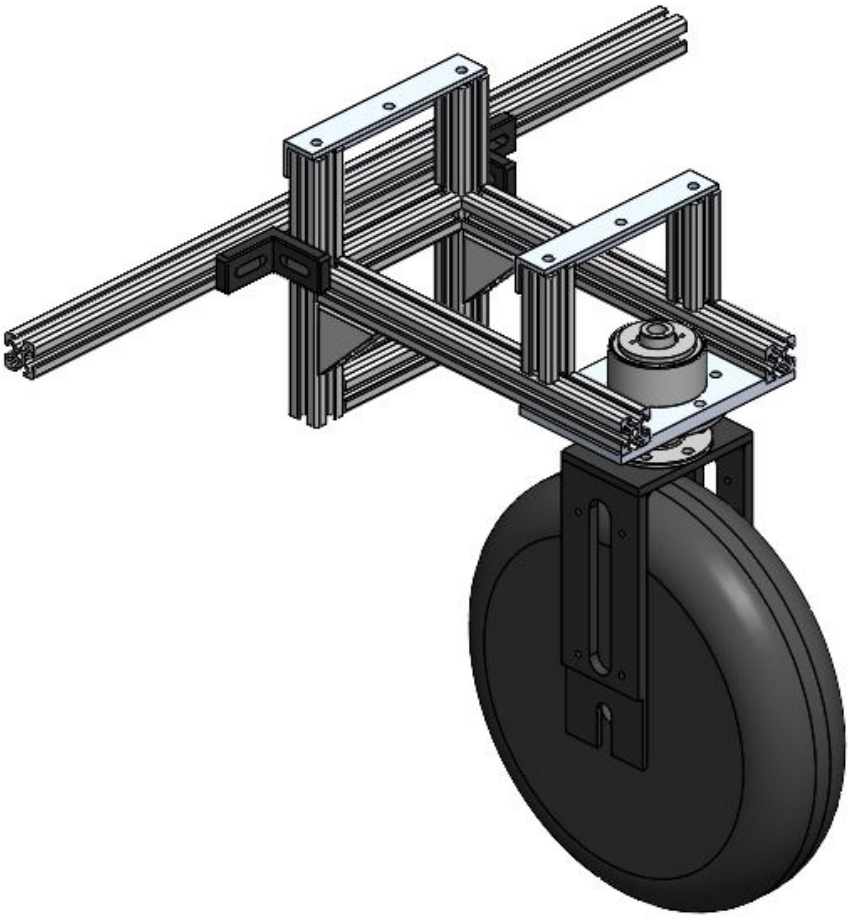
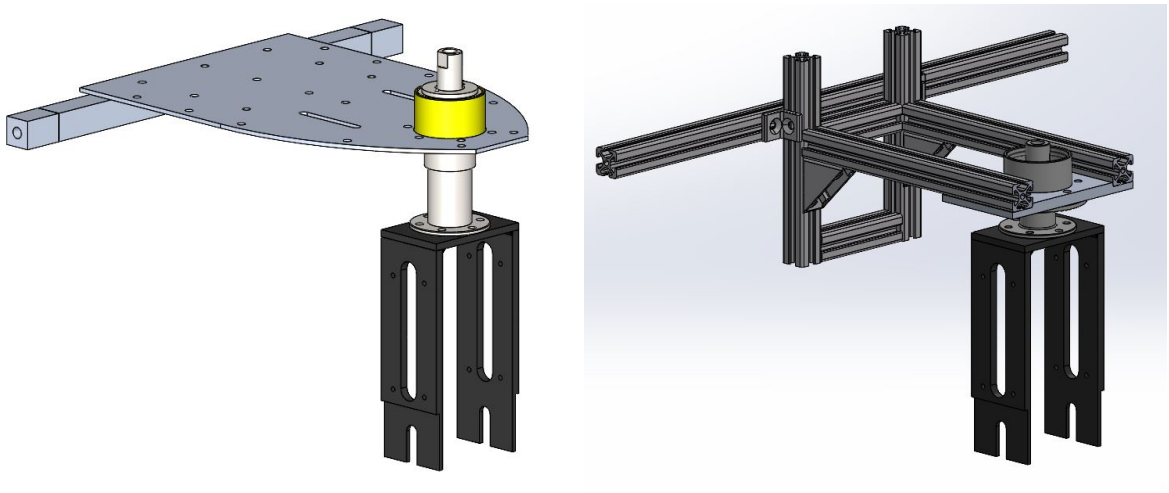


Figura 2.10: Rappresentazione completa del telaio del dispositivo

2.3 Simulazioni strutturali

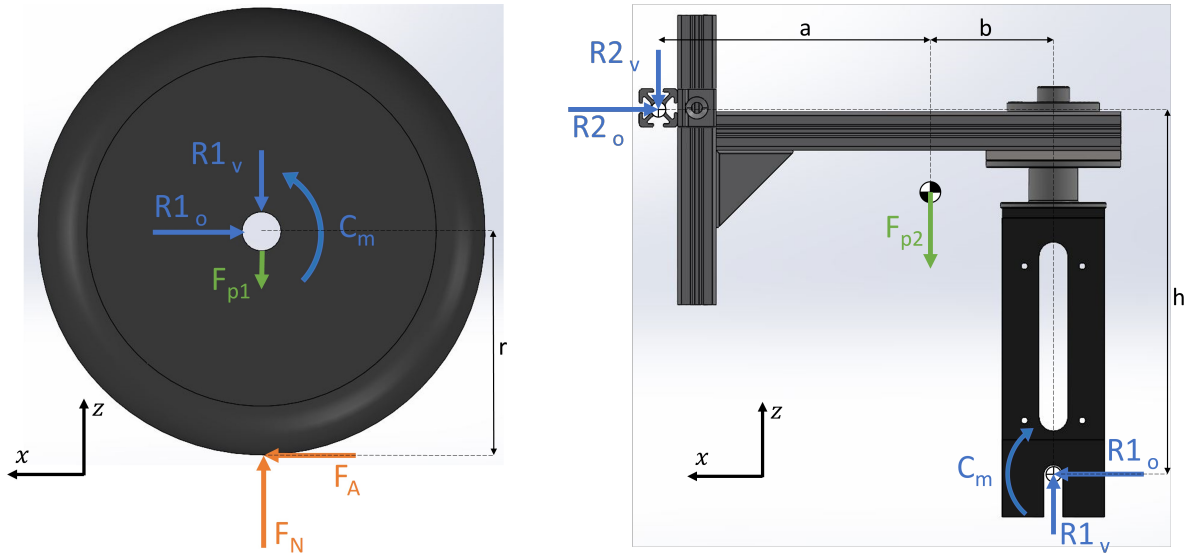
Per valutare le tensioni e gli spostamenti della struttura sono state svolte delle simulazioni strutturali sia sulla nuova soluzione, sia su quella precedente per poter ottenere un riscontro quantitativo delle modifiche apportate. Per queste simulazioni è stato utilizzato il software Ansys Workbench, che permette di modellare ed effettuare analisi meccaniche anche su sistemi complessi.
Le analisi sono state effettuate su dei sistemi semplificati di entrambi i prototipi, rappresentati in figura 2.11, i quali riportano solo i componenti strutturali importanti per l'analisi.



(a) *Struttura primo prototipo* **(b)** *Nuova struttura*

Figura 2.11: Strutture utilizzate per simulazioni FEM

Per effettuare le simulazioni strutturali è necessario impostare i vincoli e le forzanti per la struttura. In questo caso specifico la struttura è vincolata all'asse posteriore della carrozzina e alla ruota motrice del dispositivo. Per il primo vincolo si considera uno di tipo *"cerniera"*, perché la struttura è libera di ruotare intorno all'asse delle ruote posteriori; il secondo, invece, si trova nella zona di collegamento tra ruota e forcella ed è di tipo *"carrello"*, in quanto il punto di fissaggio rimane sempre alla stessa altezza e consente sia la traslazione orizzontale che la rotazione attorno all'asse ruota. Inoltre, in questa interfaccia sono concentrate anche le forzanti per la struttura del dispositivo. Per il calcolo di tali forze è utile schematizzare i diagramma di corpo libero della struttura e della ruota separatamente.
I diagrammi di corpo libero rappresentati in figura 2.12 mostrano una condizione di carico statica, per cui non sono presenti le componenti dinamiche dei due sistemi. Nella figura 2.12a è riportato il diagramma di corpo libero della ruota motrice: in blu sono raffigurate le forze di vincolo esercitate sulla ruota, in arancione le forze di contatto con il terreno (forza normale e forza di attrito) e in verde la forza peso della ruota. Nella figura 2.12b è visibile il diagramma di corpo libero della struttura con in blu le reazioni vincolari, sia nel punto di attacco all'asse della carrozzina, sia nel punto di attacco della ruota e in verde la forza peso della struttura.
Da notare che le forzanti utilizzate nelle simulazioni sono la forza $R1_o$ e la coppia motrice C_m.



(a) *Diagramma corpo libero della ruota* **(b)** *Diagramma corpo libero della struttura in vista laterale*

Figura 2.12: Diagrammi di corpo libero in configurazione di spinta longitudinale

Per completezza vengono di seguito riportate le equazioni risolventi associate al sistema ruota (eq. 2.1) e alla struttura meccanica (eq. 2.2)

$$\begin{cases} F_A - R1_o = 0 \\ F_N - R1_v - F_{p1} = 0 \\ C_m - R1_o \cdot r = 0 \end{cases} \tag{2.1}$$

$$\begin{cases} R1_o - R2_o = 0 \\ R1_v - R2_v - F_{p2} = 0 \\ R1_v \cdot (a+b) - F_{p2} \cdot a - R1_o \cdot h - C_m = 0 \end{cases} \tag{2.2}$$

Dall'esperienza precedente si è notato che la struttura subisse le maggiori sollecitazioni nel caso di sterzata. Per questa ragione si è deciso di effettuare simulazioni anche in questa configurazione e a tal proposito è stato riportato il diagramma di corpo libero in figura 2.13, in cui si fa riferimento alla sola struttura, dal momento che il diagramma di corpo libero della ruota resta uguale al caso precedente (fig. 2.12a). Anche per questo caso sono state riportate le equazioni risolventi relative al sistema (eq. 2.3).

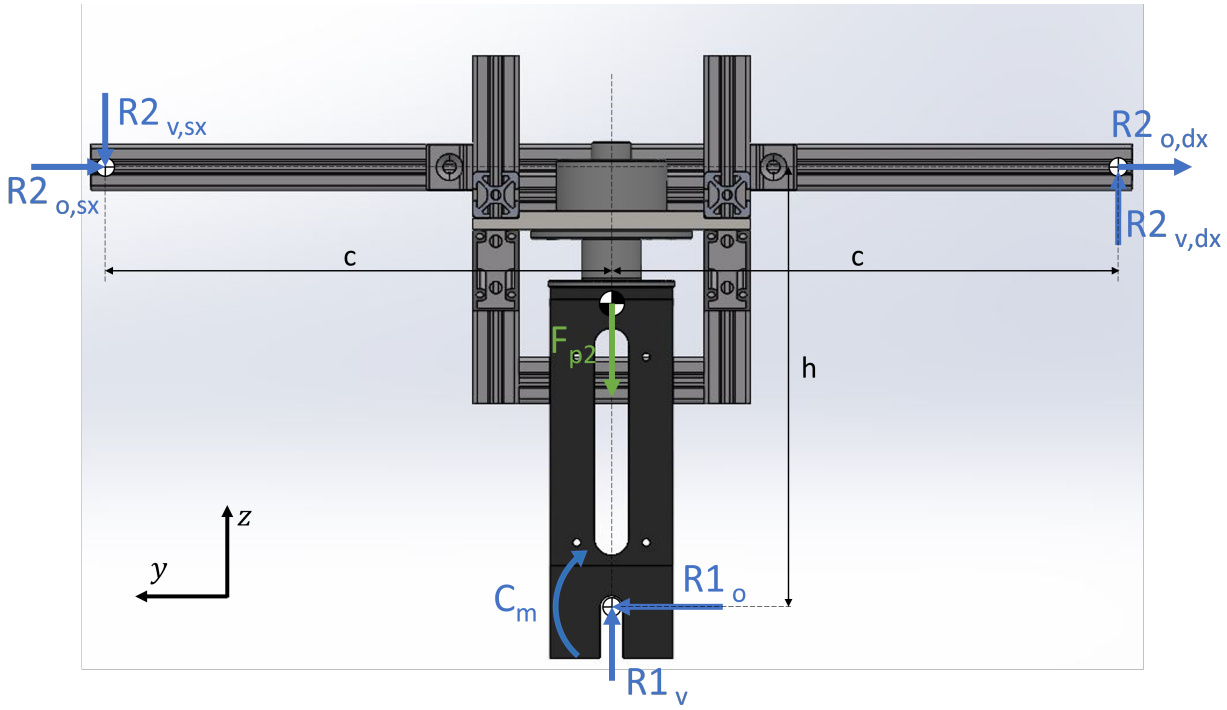


Figura 2.13: Diagramma di corpo libero nella configurazione di ruota sterzata a 90°

$$\begin{cases} R2_{o,dx} + R2_{o,sx} - R1_o = 0 \\ R1_v + R2_{v,dx} - R2_{v,sx} - F_{p2} = 0 \\ R1_v \cdot c - F_{p2} \cdot c + R2_{v,dx} \cdot 2c - C_m = 0 \\ R1_v \cdot (a+b) - F_{p2} \cdot a = 0 \end{cases} \tag{2.3}$$

Per le simulazioni in condizioni di sterzata vengono riportate le reazioni vincolari alla carrozzina come segue:

$$\begin{cases} R2_{o,dx} + R2_{o,sx} = R2_o \\ R2_{v,dx} - R2_{v,sx} = R2_v \end{cases} \tag{2.4}$$

Dove:

- ***F_A*** *è la forza di attrito tra ruota e terreno;*
- ***F_N*** *è la forza normale tra ruota e terreno;*
- ***F_{p1}*** *e* ***F_{p2}*** *sono la forza peso rispettivamente della ruota e della struttura;*
- ***C_m*** C_m *è la coppia motrice;*
- ***$R1_o$*** *e* ***$R1_v$*** *sono le forze di reazione vincolare tra la ruota e la struttura;*
- ***$R2_o$*** *e* ***$R2_v$*** *sono le forze di reazione vincolare tra la struttura e la carrozzina;*
- ***r*** *è il raggio della ruota;*
- ***h*** *è la distanza verticale tra l'asse della ruota e quello del profilo di collegamento alla carrozzina;*

- ***a*** *e* ***b*** *sono le distanze orizzontali rispetto al baricentro della struttura rispettivamente dell'asse del profilo di collegamento alla carrozzina e dell'asse della ruota.*
- ***c*** *è la distanza tra i vincoli della carrozzina e l'asse dell'albero di rotazione della struttura.*

Di seguito nella tabella 2.2 vengono riportati i dati geometrici delle due strutture.

Dati	Struttura precedente	Nuova struttura
r	110 mm	110 mm
h	190 mm	180 mm
a	157,5 mm	135 mm
b	77,5 mm	80 mm
c	230 mm	230 mm

Tabella 2.2: Dati geometrici strutture

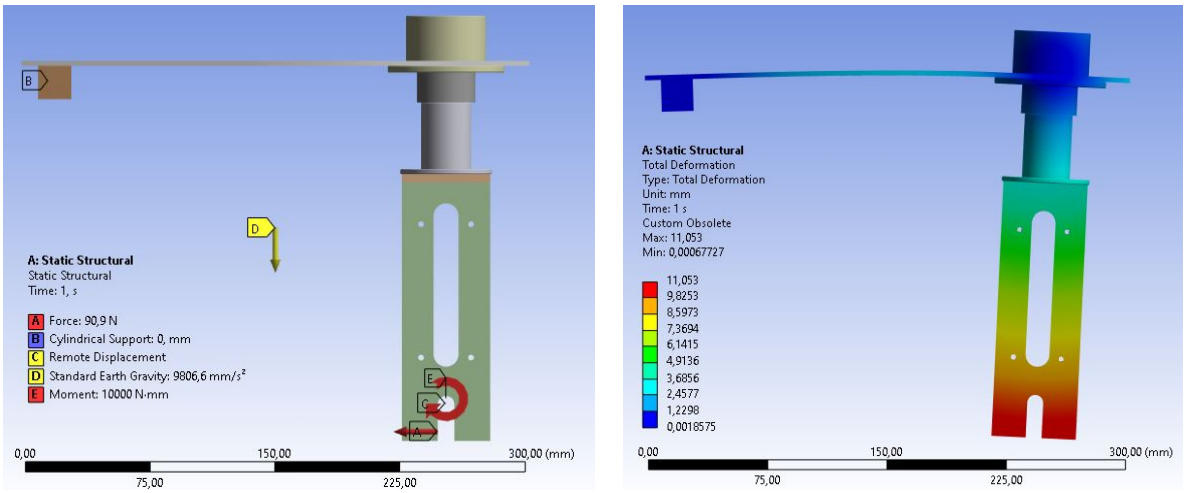
Come detto in precedenza, per le simulazioni strutturali bisogna inserire i valori delle due forzanti per la struttura: nello specifico si è scelto di utilizzare la coppia motrice C_m = 10 Nm. Di conseguenza è necessario calcolare la reazione vincolare orizzontale $R1_o$ risolvendo il sistema di equazioni 2.1.
In particolare, risolvendo l'ultima equazione è noto che la reazione vincolare è pari a:

$$R1_o = \frac{C_m}{r} = 90,9N \tag{2.5}$$

Sono state eseguite due simulazioni differenti per ogni tipo di struttura: nella prima si è simulata la configurazione di spinta longitudinale e nella seconda la configurazione di ruota sterzata a 90°. Per ogni simulazione si è riportato uno schema in cui sono rappresentati i vincoli e le forzanti, ossia i parametri che è necessario impostare nella simulazione, e uno schema raffigurante la deformata con scala 1:1.
Nei successivi paragrafi si mostrano i risultati ottenuti dalle simulazioni strutturali realizzate, presentando per ogni caso entrambi gli schemi citati in precedenza. Inoltre, vengono riportati in tabella i risultati rilevanti.

2.3.1 Simulazione 1: struttura del primo prototipo, spinta longitudinale

La prima simulazione strutturale prevede l'analisi della struttura del primo prototipo nella configurazione di spinta longitudinale. In figura 2.14 sono rappresentati rispettivamente i parametri impostati per la simulazione (fig. 2.14a) e la struttura deformata (fig. 2.14b). Vengono riportati anche i risultati più significativi (tab. 2.3).



(a) *Rappresentazione dei vincoli e delle forzanti* **(b)** *Rappresentazione della struttura deformata*

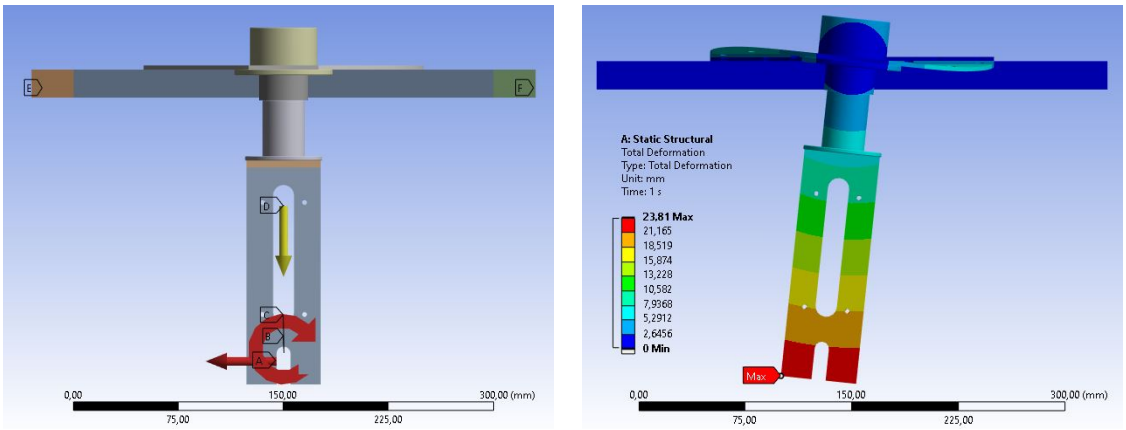
Figura 2.14: Simulazione strutturale 1

Caratteristica	Valore
$R1_v$	137,06 N
$R2_o$	90,9 N
$R2_v$	115,55 N
Deformazione massima	11,05 mm
Sollecitazione massima	160,96 Mpa

Tabella 2.3: Risultati simulazione 1

2.3.2 Simulazione 2: struttura del primo prototipo, sterzata 90°

In questo caso è stata simulata la configurazione di ruota sterzata a 90°. In figura 2.15 è visibile lo schema rispettivamente dei parametri impostati e della deformata. In tabella 2.4 si mostrano i risultati ottenuti. È possibile notare come in questo secondo caso simulato la struttura risulti più sollecitata rispetto al primo. Infatti, si ottiene una deformazione pari a circa il doppio rispetto al caso precedente.



(a) *Rappresentazione dei vincoli e delle forzanti* **(b)** *Rappresentazione della struttura deformata*

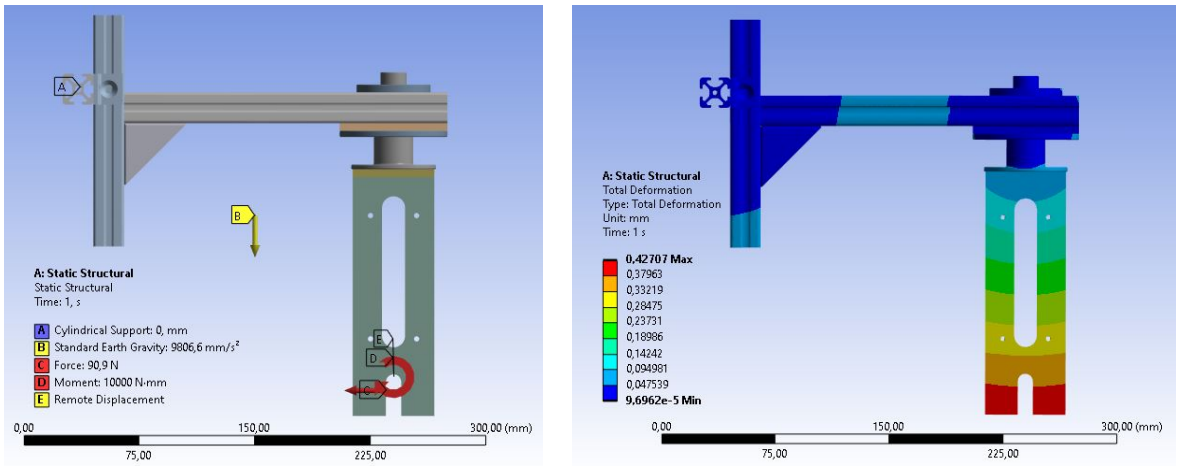
Figura 2.15: Simulazione strutturale 2

Caratteristica	Valore
$R1_v$	15,42 N
$R2_o$	90,9 N
$R2_v$	6,1 N
Deformazione massima	23,81 mm
Sollecitazione massima	188,67 Mpa

Tabella 2.4: Risultati simulazione 2

2.3.3 Simulazione 3: nuova struttura, spinta longitudinale

In questo caso è stato simulato il comportamento della nuova struttura nella configurazione di spinta longitudinale, di cui vengono riportati i parametri impostati e la deformata risultante in figura 2.16. In tabella 2.5 si mostrano i risultati significativi. Si fa notare che per questa simulazione la scala colore rappresenta dei valori di deformazione che si differenziano notevolmente da quelli presenti nelle scale colore delle due simulazioni precedenti. Nello specifico, in questo caso il valore di deformazione massima risulta molto minore rispetto ai casi precedenti, evidenza del fatto che la struttura risulta più rigida.



(a) *Rappresentazione dei vincoli e delle forzanti* **(b)** *Rappresentazione della struttura deformata*

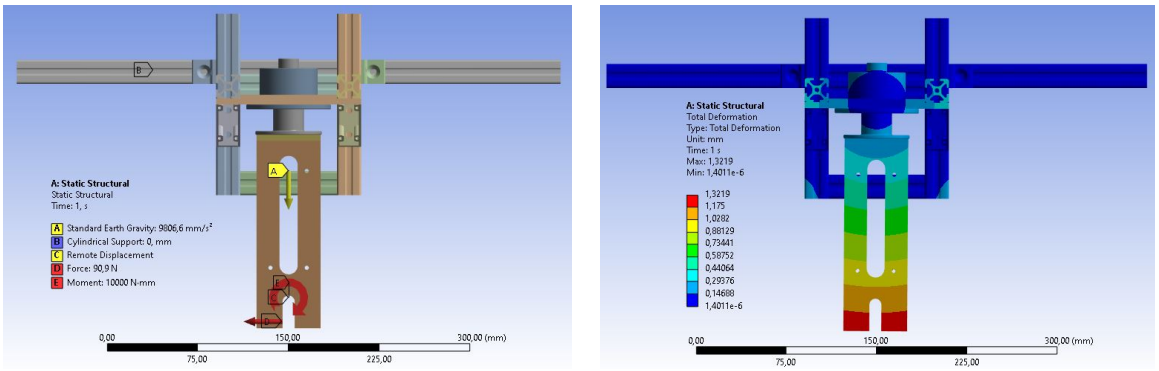
Figura 2.16: Simulazione strutturale 3

Caratteristica	Valore
$R1_v$	159,67 N
$R2_o$	90,9 N
$R2_v$	137,94 N
Deformazione massima	0,43 mm
Sollecitazione massima	86,59 Mpa

Tabella 2.5: Risultati simulazione 3

2.3.4 Simulazione 4: nuova struttura, sterzata 90°

In questo caso è stato simulato il comportamento della struttura rispetto nella configurazione di ruota sterzata a 90° rappresentando in figura 2.17 i due schemi relativi ai parametri impostati e la deformata ottenuta. In tabella 2.6, come per i casi precedenti, si riportano i risultati numerici ottenuti. Da quest'ultimi si osserva che anche per la nuova struttura la configurazione di sterzata è quella in cui si riscontrano le sollecitazioni e le deformazioni maggiori. Ciò nonostante, rispetto alla struttura del primo prototipo, queste risultano minori evidenziando un notevole miglioramento in termini di rigidità.



(a) *Rappresentazione dei vincoli e delle forzanti* **(b)** *Rappresentazione della struttura deformata*

Figura 2.17: Simulazione strutturale 4

Caratteristica	Valore
$R1_v$	13,58 N
$R2_o$	90,9 N
$R2_v$	8,15 N
Deformazione massima	1,32 mm
Sollecitazione massima	125,72 Mpa

Tabella 2.6: Risultati simulazione 4

In tabella 2.7 vengono riportati i valori di deformazione massima della struttura, rispettivi alle simulazioni. Si può osservare che la nuova struttura presenta una riduzione di circa il 95% del valore di deformazione massima.

	Deformazione massima
Simulazione 1	11,05 mm
Simulazione 2	23,81 mm
Simulazione 3	0,43 mm
Simulazione 4	1,32 mm

Tabella 2.7: Tabella riassuntiva deformazioni massime

2.4 Design della scocca

In questo paragrafo viene illustrata la soluzione progettuale ideata per la copertura della motoruota, mostrata in figura 2.18. Il modello si propone di garantire protezione ai componenti elettronici rispetto alle impurità esterne e allo stesso tempo conferire al dispositivo un design migliore.

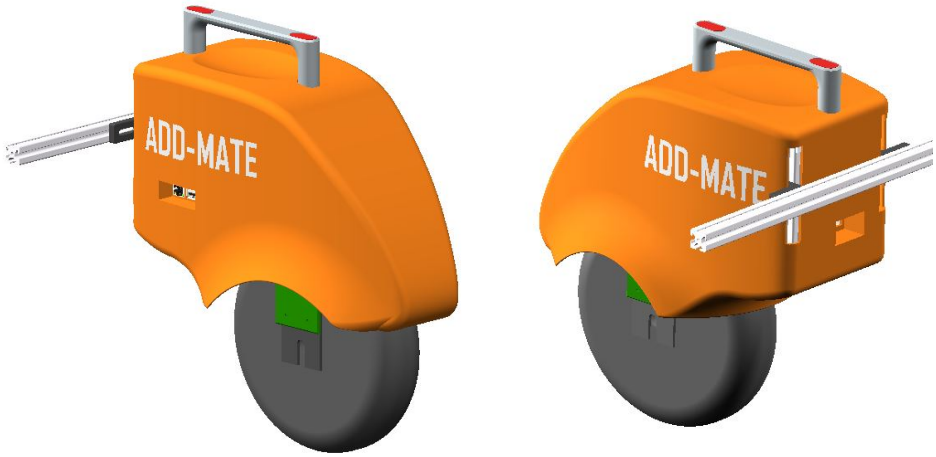
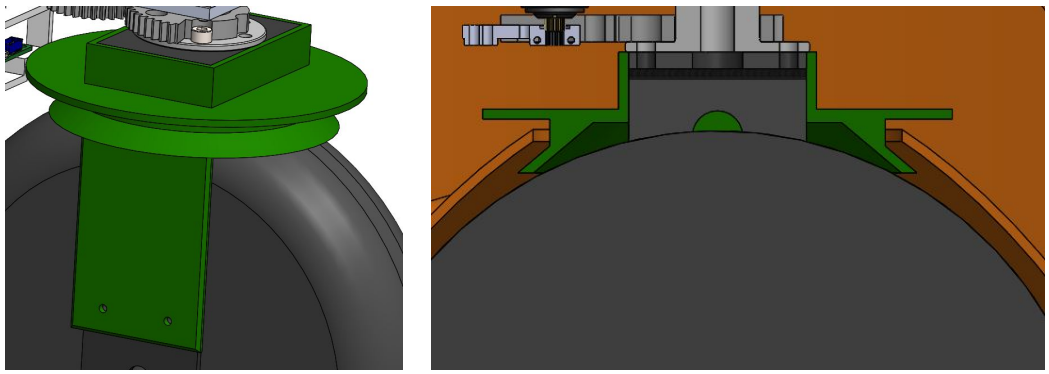


Figura 2.18: Rappresentazione del dispositivo completo di scocca

Si intende adottare l'additive manufacturing come tecnica di produzione per questi componenti, in quanto consente la realizzazione di geometrie complesse. Si è resa poi necessaria anche la progettazione di un altro tipo di copertura (fig. 2.19a) solidale alla forcella, al fine di impedire il passaggio di polveri all'interno del dispositivo. A tal scopo il modello ideato, nel momento in cui viene montato insieme alla copertura della motoruota, forma un labirinto che ostacola il passaggio alle polveri provenienti dall'esterno, come è visibile nella figura 2.19b.



(a) *Rappresentazione del montaggio* **(b)** *Dettaglio della soluzione a labirinto*

Figura 2.19: Carter forcella

Le dimensioni di ingombro massimo del dispositivo così progettato risultano di 0,35 x 0,21 x 0,41 m.

Capitolo 3

SISTEMA MECCATRONICO

Nel capitolo seguente sono descritti i vari componenti che costituiscono il sistema meccatronico utilizzato. In seguito viene introdotto il funzionamento del motore con il driver e la tipologia di comunicazione tra quest'ultimo e il microprocessore. Successivamente, sono riportati i risultati delle prove sperimentali eseguite preliminarmente, in controllo velocità, per la scelta dei parametri del controllore, e in controllo coppia, per la caratterizzazione del motore. Infine, viene presentato il test grazie al quale si analizza il comportamento del sistema in caso di frenata.

3.1 Descrizione sistema

Il sistema meccatronico complessivo è composto da una ruota per e-scooter che al suo interno presenta un motore brushless da 250 W, un driver del motore SOLO MINI V2, un microprocessore Rock Pi 4, un servomotore SunFounder 20 kg, un sensore di orientamento IMU, otto celle di carico e due schede di comunicazione bluetooth ESP32.
Nella figura 3.1 sono raffigurati tutti i componenti sopra elencati con i relativi collegamenti.

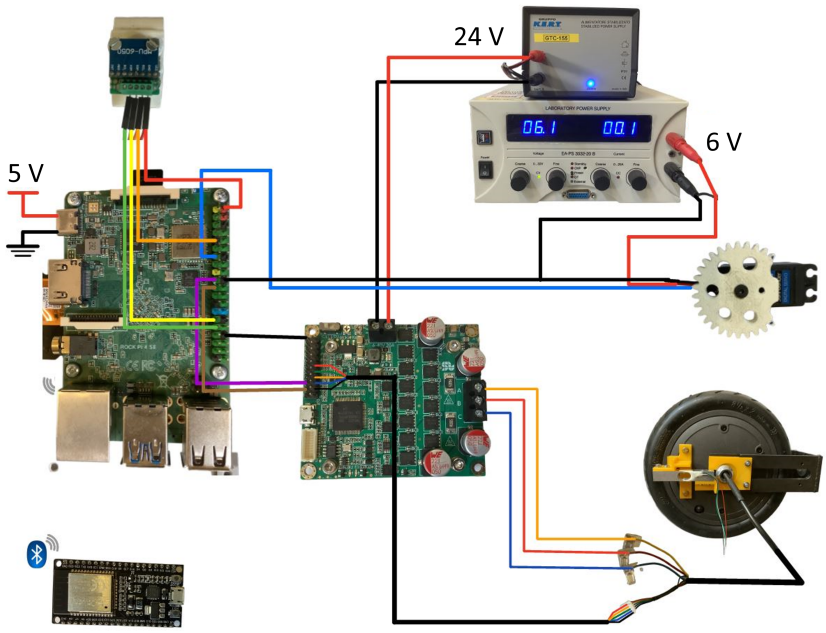


Figura 3.1: Collegamenti del sistema meccatronico

In tabella 3.1 sono specificati i collegamenti da effettuare con ilSBC Rock pi per permettere il funzionamento del sistema.

Rock Pi	PIN 2 (5V) PIN 9 (GND) PIN 29 (I2C6_SCL) PIN 31 (I2C6_SDA)	VCC GND SCL SDA	**MPU 6050**
	PIN 13 (PWM1) PIN 20 (GND)	CMD GND	**Servo**
	PIN 19 (UART4_TXD) PIN 21 (UART4_RXD) PIN 20 (GND)	RX TX GND	**SOLO**

Tabella 3.1: Collegamenti Rock Pi con i vari dispositivi

In tabella 3.2 sono indicati i collegamenti tra il driver SOLO e il motore di spinta.

SOLO	**BLDC**
5V	Hall VCC
EA	Hall A
EB	Hall B
EC	Hall C
GND	Hall GND
A	Fase A
B	Fase B
C	Fase C

Tabella 3.2: Collegamenti tra il driver SOLO e la motoruota - I colori rappresentati sono associati al colore dei cavi di collegamento del motore.

3.1.1 Descrizione motoruota

Nella figura 3.2 è osservabile la ruota motorizzata per e-scooter utilizzata. Al suo interno è integrato un motore di tipo brushless a corrente continua, diffusamente utilizzato in quanto molto efficiente e affidabile.
Il motore brushless è caratterizzato dal fatto di non utilizzare le spazzole, e più precisamente la combinazione spazzole-collettore, per effettuare la commutazione delle correnti in funzione della posizione dell'elemento rotore, bensì utilizza opportuni dispositivi di elettronica di potenza. Questo tipo di motori presenta avvolgimenti elettrici solamente a statore, mentre sull'elemento rotore sono montati dei magneti permanenti che generano un campo di induzione magnetica solidale con il rotore. Quando quest'ultimo è messo in rotazione, l'alternanza tra poli positivi e poli negativi fa sì che gli avvolgimenti elettrici a statore vedano un campo magnetico variabile. Questi dispositivi si differenziano dai motori con spazzola per essere a tutti gli effetti dei dispositivi a più fasi. In ogni fase è generata una forza contro-elettromotrice calcolabile per mezzo della legge di Faraday in funzione della velocità di variazione del campo di induzione magnetica, del numero di spire appartenenti alla singola fase e in funzione della lunghezza degli avvolgimenti. Il campo di induzione magnetica dipende dall'angolo elettrico, ossia dalla posizione occupata dal rotore in un preciso istante temporale. Quest'ultima informazione viene fornita da specifici sensori presenti all'interno del motore; in questo si tratta di sensori ad effetto Hall. In particolare, il feedback sulla posizione del rotore fornito da sensori viene utilizzato da un microprocessore per determinare la tensione da applicare ai fili dello statore e quindi controllare la commutazione della corrente (commutazione elettronica). In base, quindi, alla configurazione dei magneti permanenti e alla commutazione elettronica, il rotore inizia a girare e la commutazione delle fasi dello statore permette al motore brushless di generare una rotazione fluida e precisa. I motori brushless presentano diversi vantaggi rispetto ai motori con spazzola, tra i quali si distinguono per avere una dimensione estremamente compatta, per essere più leggeri, oltre che per evitare numerosi inconvenienti legati alla presenza delle spazzole, come l'usura, la possibilità di generare scintille in prossimità di materiali infiammabili e l'aumento della dimensione assiale del motore, il che giustifica il loro largo impiego nella maggior parte dei servosistemi elettromeccanici [17].

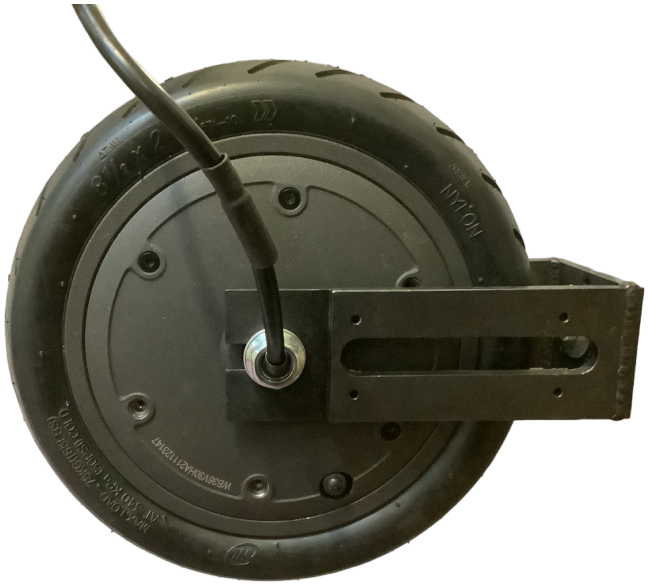


Figura 3.2: Foto della motoruota

3.1.2 Descrizione microprocessore

Per il controllo del sistema è stata utilizzata una scheda Single-Board Computer (SBC) Rock Pi 4 prodotta da Radxa [18], progettata per scopi di prototipazione e sviluppo. Rock Pi 4 presenta le seguenti caratteristiche e componenti:

- processore basato su architettura ARM con sei core, il RK3399;
- RAM LPDDR4 da 4 Gb;
- supporto per l'archiviazione tramite slot per schede microSD e connettore M.2 per SSD NVMe, che consente l'espansione dello storage in base alle esigenze;
- vasta gamma di opzioni di connettività, tra cui Wi-Fi 802.11ac, Bluetooth 5.0, Gigabit Ethernet, e diverse porte USB (USB 3.0 e USB 2.0);
- 40 pin GPIO (General-Purpose Input/Output) che consentono di collegare sensori e altri dispositivi;
- possibilità di supportare diversi sistemi operativi Linux, come Debian e Ubuntu, oltre che molte distribuzioni Linux personalizzate;
- dimensioni compatte, che lo rendono facilmente integrabile in progetti embedded o personalizzati.

Nello specifico per questa applicazione è stata scelta la SBC Rock PI 4 SE riportata in figura 3.3.

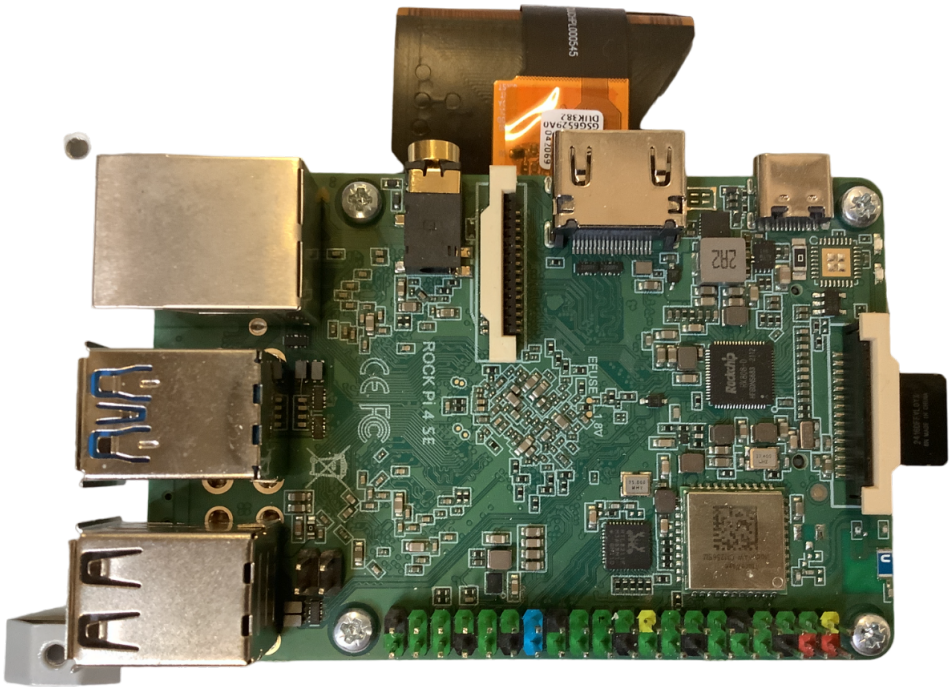


Figura 3.3: Rock PI 4 SE

3.1.3 Descrizione driver

Il driver utilizzato per il motore della ruota è il SOLO MINI V2 [19], riportato in figura 3.4. Esso presenta la peculiarità di poter controllare diversi tipi di motori, tra cui motori Direct Current (DC), Brushless Direct Current (BLDC),Permanent Magnet Synchronous Motor (PMSM) e Alternative Current Induction Motor (ACIM). Inoltre, permette la comunicazione tramite USB, UART e CANopen e dispone della capacità di auto-tuning e identificazione automatica dei parametri del motore. Permette il controllo sia sensorless, sia basato su sensori, per cui supporta ingressi da encoder e sensori Hall. Si possono effettuare controlli in anello chiuso e in anello aperto ed è supportato il controllo di coppia, velocità e posizione. È caratterizzato da un ampio intervallo di tensione di ingresso, da 6 V a 40 V, e da una corrente in uscita continua di 20 A fino a un massimo di 80 A. Permette il funzionamento del motore come freno rigenerativo bidirezionale, limitato alla corrente massima, e presenta un dissipatore montato sul retro per migliorare il comportamento termico. Il driver è stato utilizzato in controllo velocità impostandone i parametri del controllore (k_p e k_i) e la velocità. Per il controllo coppia, invece, è necessario impostare un valore per la corrente i_q. Per comprendere a cosa corrisponde questa corrente bisogna specificare che questo driver è basato sulla tecnica FOC, ossia una tecnica di controllo utilizzata in elettronica di potenza e nell'automazione per il controllo preciso della velocità e della coppia dei motori elettrici. Il FOC mira a controllare il flusso magnetico nel motore in modo da ottenere una performance ottimale, garantendo una maggiore efficienza energetica da parte del motore, una migliore gestione del comportamento termico del motore e una maggiore affidabilità. Piuttosto che controllare direttamente la corrente nelle fasi del motore, il FOC controlla il flusso magnetico all'interno del motore, variando il valore della corrente magnetizzante i_q, detta anche corrente di quadratura. La variazione del valore di questa corrente influenza la coppia esercitata dal motore [20]. Nello specifico è stato scelto questo driver in quanto consente il controllo del motore BLDC con sensori Hall descritto nel paragrafo 3.1.1 e in particolare, a differenza del driver presente nel primo prototipo, permette il controllo di coppia del motore e la frenata rigenerativa. Inoltre, essendo molto più reattivo nel controllo del motore, permette il funzionamento del sistema a frequenze più elevate.

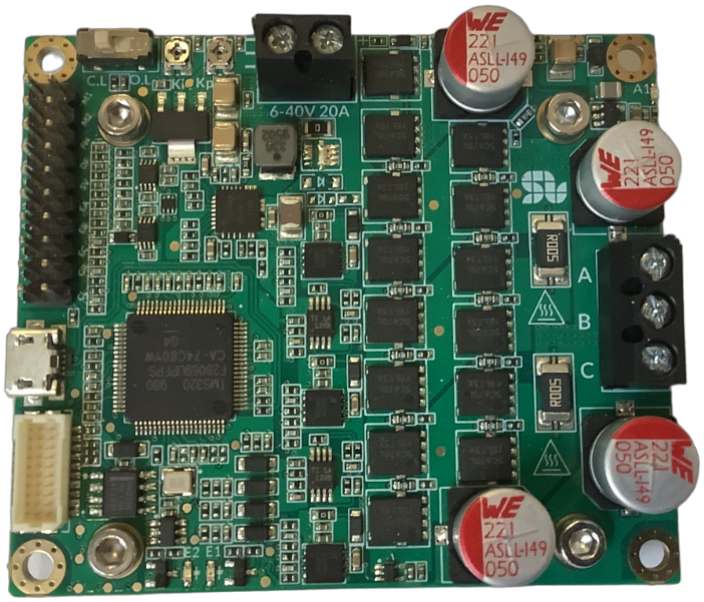


Figura 3.4: SOLO MINI V2

3.1.4 Descrizione del servomotore

Il servomotore è stato utilizzato per eseguire la rotazione della forcella e quindi permettere l'assistenza anche in sterzata da parte del dispositivo. Nello specifico è stato utilizzato un servomotore SunFounder SF3218MG [21] mostrato in figura 3.5. Le caratteristiche principali di questo dispositivo consistono nella possibilità di applicare una rotazione con un angolo di 270°, una coppia massima di 20,5 kg·cm e una velocità di 0,18 s/60°. Il servomotore presenta una larghezza di banda di controllo di 2000 µs con una larghezza di banda morta di 5 µs, che lo rende molto preciso.

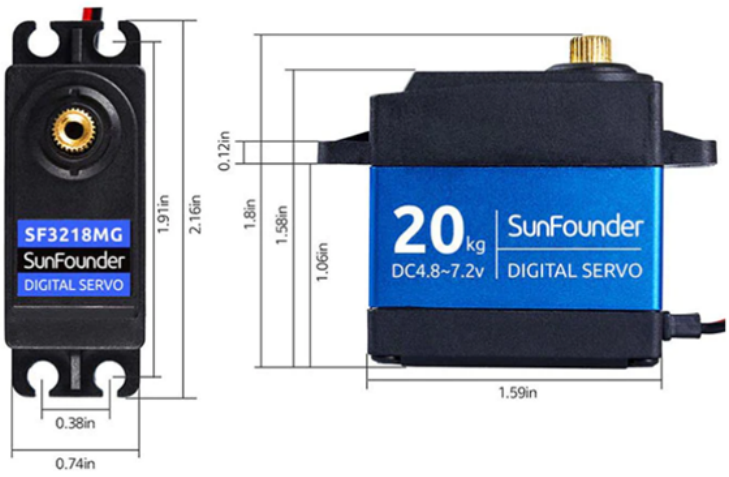


Figura 3.5: SunFounder SF3218MG

In figura 3.6 è possibile osservare il montaggio del servomotore all'interno del dispositivo. Si può notare come la ruota dentata montata sul motore si accoppi con una seconda ruota dentata montata sulla forcella, coassiale con l'albero verticale del dispositivo. In questo modo una variazione dell'angolo del SunFounder comporta una corrispondente variazione di direzione della ruota del dispositivo di un angolo corrispondente.

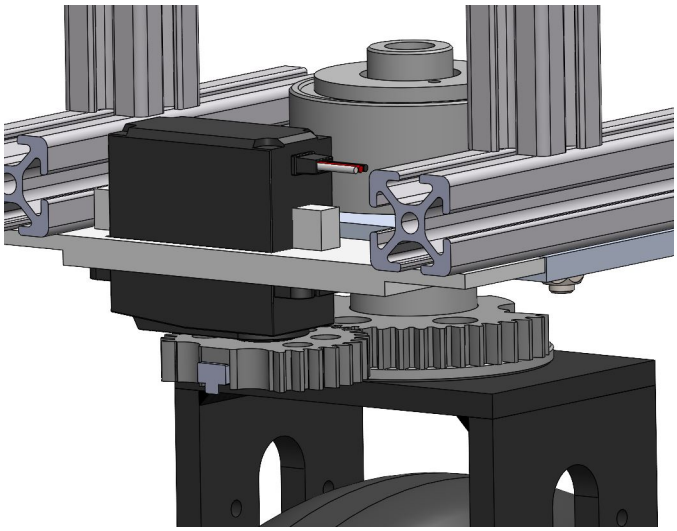


Figura 3.6: Montaggio servomotore

3.1.5 Descrizione dei sensori

Per poter misurare la velocità e l'inclinazione della carrozzina è stato montato su di essa un sensore Inertial Measurement Unit (IMU): nello specifico si tratta del sensore MPU 6050 visibile nella figura 3.7, che combina un accelerometro e un giroscopio in un unico modulo. L'accelerometro all'interno del MPU-6050 misura l'accelerazione lineare lungo i tre assi spaziali x, y e z, solidali al sensore stesso, mentre il giroscopio misura la velocità angolare di rotazione intorno agli stessi assi. Da queste misure è possibile derivare i valori di altre grandezze. Il sensore MPU-6050 include un processore integrato che elabora i dati provenienti dall'accelerometro e dal giroscopio per calcolare l'orientazione e il movimento. Il sensore presente nella figura 3.7 è quello montato sul telaio della MWC, allo scopo di misurare gli angoli di inclinazione della carrozzina.

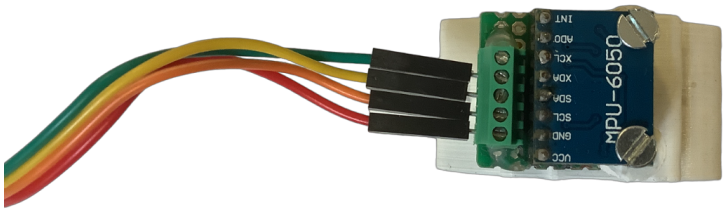


Figura 3.7: Inertial Measurement Unit: MPU 6050

Come descritto in precedenza, l'input di comando al dispositivo è impartito dalla spinta sui pushrim. Dalla figura 3.8 è possibile osservare come sulla ruota siano montati diversi sensori. Nello specifico, sull'asse di ogni ruota è stato montato un modulo MPU 6050 descritto in precedenza, utile per monitorare la velocità angolare di ogni singola ruota.

Figura 3.8: Ruota sensorizzata

Inoltre, si può notare come tra pushrim e ruota sono montate quattro celle di carico al fine di misurare la forza di spinta esercitata dall'utente. In figura 3.9 è possibile osservare un dettaglio del montaggio. Le celle di carico sono disposte radialmente per essere sensibili solo alla componente tangenziale della forza esercitata dall'utente, che è la componente efficace per la trasmissione del moto.

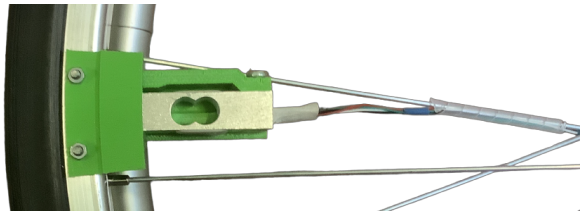


Figura 3.9: Dettaglio di montaggio della cella di carico

Il sensore IMU e le celle di carico di ogni ruota sono collegate ad una scheda ESP32 [22], ossia una piattaforma di sviluppo basata su un microprocessore a 32 bit con doppio core, che offre avanzate capacità di connettività wireless (Wi-Fi e Bluetooth) e 36 porte GPIO per collegare dispositivi esterni. In figura 3.10 è possibile osservare un dettaglio della scheda montata sulla ruota.

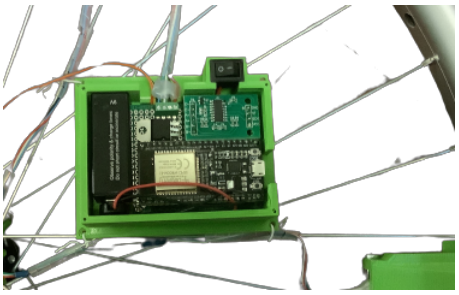


Figura 3.10: Dettaglio dell'elettronica ruota sensorizzata

Per completezza viene mostrata in figura 3.11 la carrozzina utilizzata per i test su cui sono state montate le ruote sensorizzate.

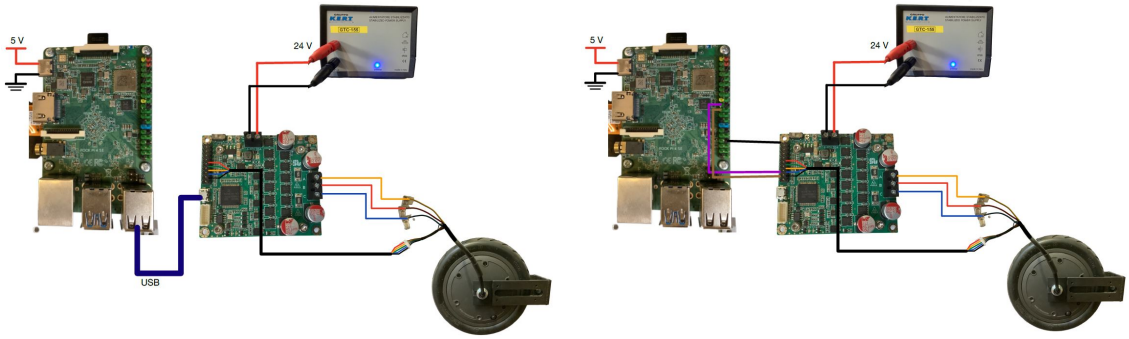
Figura 3.11: Carrozzina manuale con ruote sensorizzate

3.2 Setup e calibrazione dei sensori Hall

Come detto in precedenza, il motore utilizzato è un motore di tipo brushless a corrente continua con sensori Hall, il cui controllo necessita il collegamento dei sensori Hall sui pin specifici e quello delle tre fasi del motore. Prima di poterlo utilizzare, però, bisogna impostare il setup ed effettuare la calibrazione dei sensori. Nello specifico, il setup consiste nell'effettuare il corretto collegamento tra le tre fasi del motore e i corrispondenti sensori Hall. Per far ciò è necessario collegare il driver a un PC tramite la porta di comunicazione USB e utilizzare la piattaforma Motion Terminal [23], messa a disposizione dall'azienda SOLO (produttrice del driver). Dal terminale, una volta configurata la porta di comunicazione, bisogna impostare alcuni parametri specifici come il numero di poli (27 in questo caso), il tipo di motore(BLDC) e anche un limite di corrente. Forniti tali parametri, tramite il terminale, si effettua l'identificazione del motore e la calibrazione dei sensori. A seguito di queste operazioni il motore è pronto al funzionamento. È da notare che la ruota ha un sistema di riferimento intrinseco, rispetto al quale il driver può comandare la rotazione in senso orario o antiorario. Testando il funzionamento del motore, data una certa direzione e valore di velocità di rotazione, possono verificarsi tre scenari: il motore non risponde al comando di rotazione; il motore ruota, ma nel verso opposto rispetto a quello impostato; il motore funziona correttamente, ovvero ruota nella direzione comandata. Nel caso in cui si verifichino i primi due scenari, è necessario invertire le fasi del motore tra di loro per trovare la combinazione corretta con i sensori Hall. Questa procedura può essere ripetuta finché il motore ruoti nella giusta direzione.
Una volta che si è accertato il corretto collegamento del driver al motore, la piattaforma Motion Terminal consente il controllo del motore stesso, permettendo di decidere il tipo di controllo, il verso di rotazione e la scelta dei parametri di funzionamento. Per lo studio del funzionamento del driver e del motore è stato impostato un controllo velocità e variando il valore dei parametri del controllore k_p e k_i si è analizzata la risposta del sistema.

3.3 Test comunicazione tra microprocessore e driver

Successivamente il driver è stato collegato alla scheda Rock Pi e la comunicazione è resa possibile tramite due porte di comunicazione, USB e UART, su diversi livelli di baud rate.In figura 3.12 sono presenti gli schemi di collegamento per entrambi i tipi di porte.
Nel caso di collegamento tramite USB, la comunicazione avviene utilizzando il livello di baud rate del driver più elevato, corrispondente a 937500 bit/s. L'unico baud rate comune al driver e al microprocessore, nel caso di collegamento tramite UART, è pari a 115200 bit/s.



(a) *Collegamento via USB* **(b)** *Collegamento via UART*

Figura 3.12: Schemi di collegamento

Per poter scegliere quale tra le due porte di comunicazione utilizzare sono stati eseguiti diversi test. Di seguito viene riportato un esempio significativo di confronto. In questo test è stato fornito come set un comando cicloidale con velocità di regime pari a 400 rpm con una frequenza di circa 50 Hz. In figura 3.13 si può osservare in blu il set teorico di comando, in giallo il feedback di velocità ottenuto con collegamento tramite USB e in rosso il feedback ottenuto dal collegamento tramite UART. Si può notare come nel feedback rosso sono presenti diversi picchi verso il basso, rappresentati dal valore -1, che indicano le disconnessioni tra i due sistemi. Osservando la figura, però, si può concludere che, nonostante la comunicazione tramite UART avvenga a un livello di baud rate più basso e presenti delle disconnessioni, questa consente comunque di ottenere una comunicazione più veloce. Per questo motivo si è scelto di proseguire con questo tipo di collegamento.

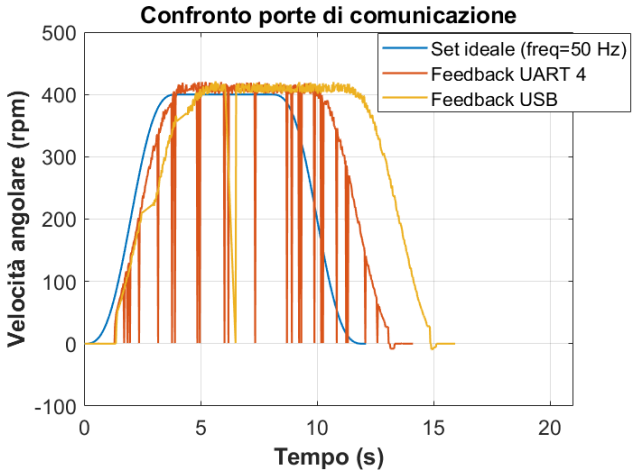


Figura 3.13: Confronto porte di comunicazione (UART - USB)

3.4 Test parametri PID

Per poter comprendere al meglio il funzionamento di driver e motore sono stati effettuati diversi test con comando in velocità, variando i parametri del controllore. Nello specifico, sono state effettuate per prime delle prove con un comando cicloidale sia nella fase di accelerazione, sia nella fase di decelerazione. Le traiettorie cicloidali permettono di ottenere un moto con accelerazione costante, evitando bruschi cambiamenti di velocità o accelerazione. In figura 3.14 si può osservare la forma del comando di set impostata per queste prime prove.

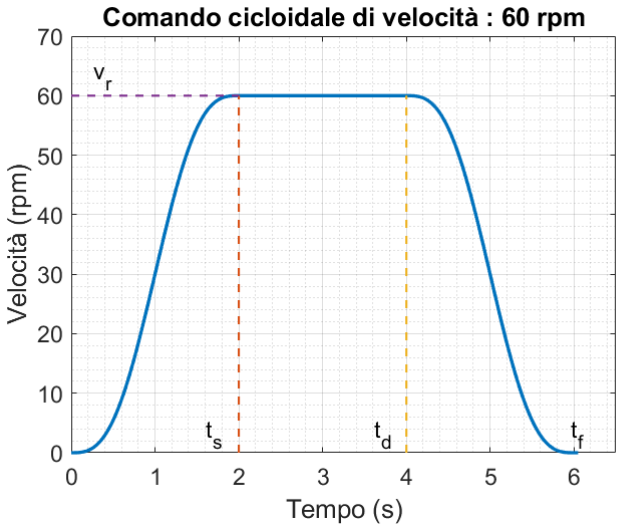


Figura 3.14: Traiettoria cicloidale utilizzata per i test

La traiettoria cicloidale è stata ottenuta applicando l'equazione 3.1

$$v(t) = \begin{cases} v_r \cdot \left(\frac{t}{t_s} - \frac{1}{2\pi} \cdot \sin\left(\frac{2\pi t}{t_s}\right)\right) & \text{per } 0 < t < t_s, \\ v_r & \text{per } t_s \leq t \leq t_d, \\ -v_r \cdot \left(\frac{t-t_d}{t_f-t_d} - \frac{1}{2\pi} \cdot \sin\left(\frac{2\pi \cdot (t-t_d)}{t_f-t_d}\right)\right) + v_r & \text{per } t_d < t < t_f. \end{cases} \tag{3.1}$$

Dove:

- $\boldsymbol{v_r}$ *indica la velocità di regime della prova (che nel caso rappresentato in figura 3.14 corrisponde a 60 rpm);*
- $\boldsymbol{t_s}$ *indica l'istante di tempo in corrispondenza del quale si raggiunge la velocità di regime;*
- $\boldsymbol{t_d}$ *indica l'istante di tempo in corrispondenza del quale la velocità inizia a diminuire;*
- $\boldsymbol{t_f}$ *indica l'istante di tempo in corrispondenza del quale si raggiunge nuovamente una velocità pari a 0 rpm.*

Per questo tipo di comando sono state eseguite diverse prove a tre velocità di regime differenti: 60 rpm, 100 rpm e 200 rpm. Per ogni velocità di rotazione sono stati fatti variare i due parametri del controllore. Nel dettaglio, per il k_p sono stati utilizzati i valori tra 0,1 e 1 con un intervallo di 0,1 e i valori 1,5 e 2, mentre per il k_i sono stati utilizzati i valori compresi tra 0,0005 e 0,003 con intervalli di 0,0005, per un totale di 72 prove per ciascuno dei tre set di velocità.

Il parametro k_p, detto anche parametro proporzionale, influenza la reattività e la velocità di risposta del sistema. Il contributo proporzionale è ottenuto dal prodotto di questo parametro per l'errore tra set e feedback. Un valore troppo alto di k_p può causare l'instabilità al sistema. Il parametro k_i, detto parametro integrativo, influenza invece la risposta a regime del sistema. Per ottenere il contributo integrativo bisogna moltiplicare il parametro per l'integrale dell'errore.
Per ogni prova sono state calcolate diverse caratteristiche: la velocità media a regime, l'errore medio a regime, la deviazione standard a regime, la presenza di overshoot, sia in salita che in discesa, e il loro valore massimo. Per poter decidere quali siano i valori ottimali dei parametri, sono state prese in considerazione le prove con un errore medio a regime minore del 3% della velocità di set, le prove che non hanno presentato un overshoot maggiore del 5% del valore di velocità di regime e quelle in cui la deviazione standard sia limitata a 1 rpm per le prove eseguite con velocità di regime di 60 rpm, a 1,4 rpm per le prove eseguite a 100 rpm e a 2,8 rpm per le prove eseguite a 200 rpm. In tabella 3.3 sono riportate le combinazioni di parametri che rispettano questi vincoli.

Velocità	60 rpm		100 rpm		200 rpm	
Parametri	k_p	k_i	k_p	k_i	k_p	k_i
	0,7	0,0015	0,8	0,0015	0,9	0,003
	1	**0,0025**	0,9	0,03	1	0,002
			1	**0,0025**	**1**	**0,0025**

Tabella 3.3: Coppie di parametri che rispettano i vincoli

In tabella 3.3 si evidenzia il fatto che la coppia di parametri, k_p =1 e k_i = 0.0025, è presente per tutti i set di velocità a cui sono state eseguite le prove. Da ciò si evince che tale combinazione di valori può essere considerata quella ottimale per comandare il motore in controllo velocità.
Di seguito vengono mostrate le prove effettuate con i seguenti parametri. Per ogni prova sono presenti due grafici: uno che ha in ascissa i campioni e uno il tempo. Su quest'ultimo è possibile valutare visivamente gli istanti di tempo durante i quali il motore si è disconnesso dal driver. Nello specifico, in figura 3.15 viene riportata la prova con una velocità di regime di 60 rpm, in figura 3.16 è rappresentata la prova eseguita a 100 rpm e infine in figura 3.17 è presente la prova effettuata con velocità di 200 rpm. Si può osservare che in tutte le prove, con i valori dei parametri k_p e k_i scelti, il motore riesca a seguire abbastanza fedelmente la legge di set. Un'altra osservazione importante è che il motore riscontra maggiori difficoltà a seguire il segnale a basse velocità.

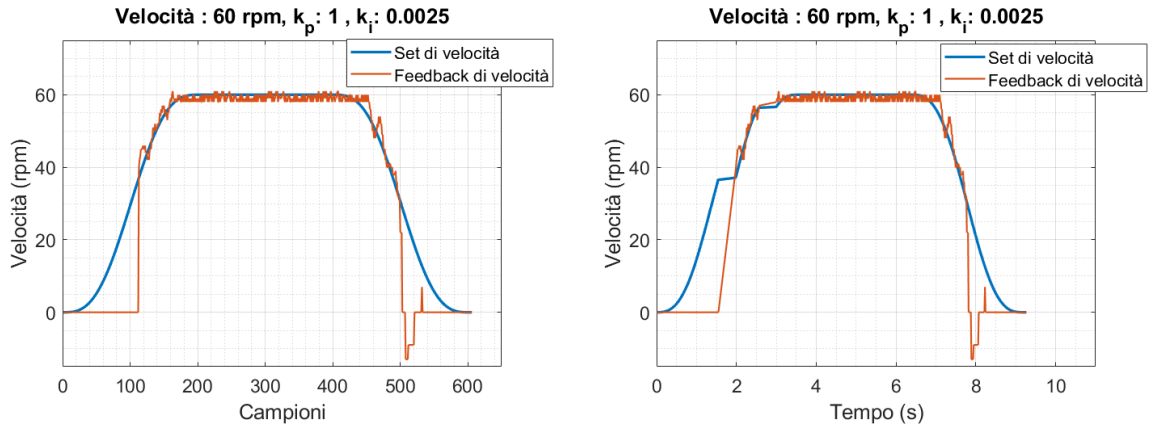


Figura 3.15: Prova comando cicloidale con velocità di regime di 60 rpm

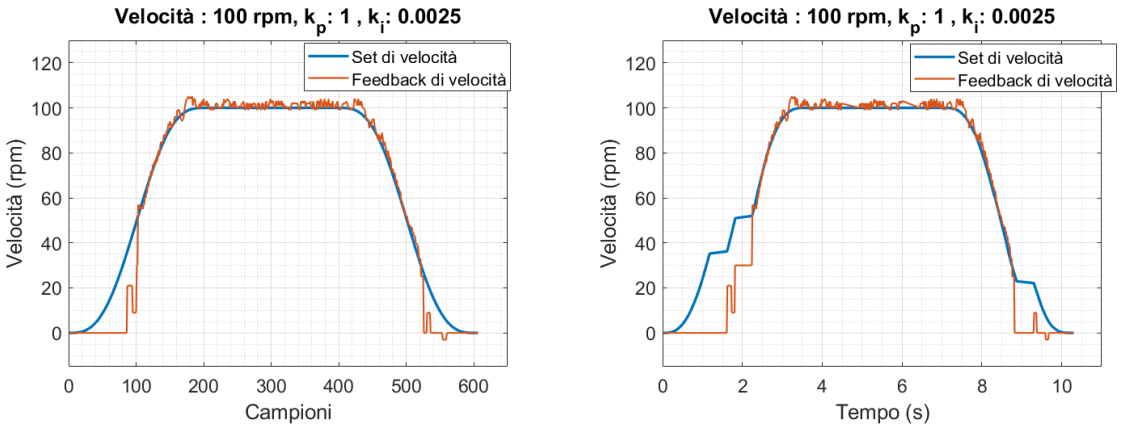


Figura 3.16: Prova comando cicloidale con velocità di regime di 100 rpm

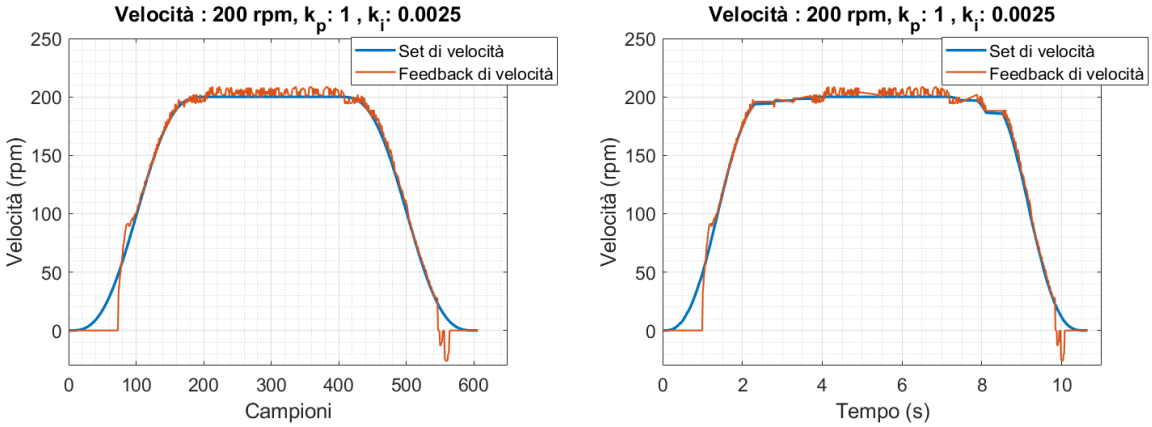


Figura 3.17: Prova comando cicloidale con velocità di regime di 200 rpm

Utilizzando gli stessi parametri sono state eseguite delle prove con comando di velocità a gradino. Anche in questo caso le prove sono state effettuate a tre velocità di rotazione differenti.
In figura 3.18 si mostra la prova effettuata con un comando di velocità a gradino di 60 rpm, dalla quale si può notare la presenza di un overshoot singolo nella fase di salita e alcuni overshoot di minore entità nella fase di discesa. In figura 3.19 è invece riportata la prova con comando di velocità a gradino di 100 rpm, che presenta una forma molto simile al caso precedente. Infine, in figura 3.20 è presente la prova con comando di velocità a gradino di 200 rpm, che rispetto ai casi precedenti mostra nella fase di salita anche un secondo piccolo overshoot.

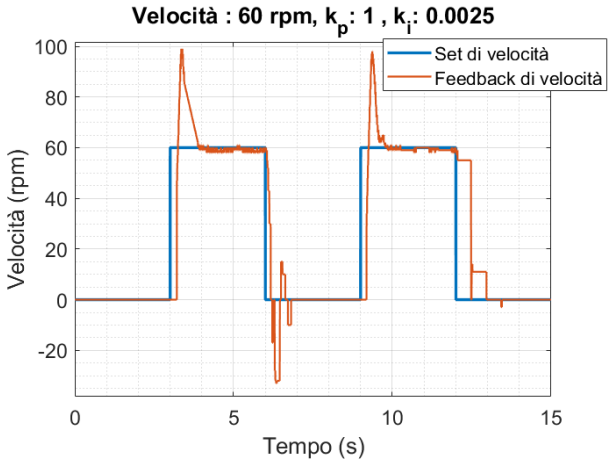


Figura 3.18: Prova comando gradino con velocità di regime di 60 rpm

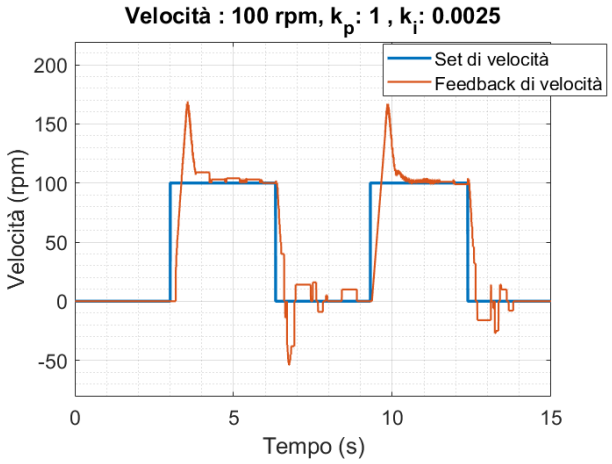


Figura 3.19: Prova comando gradino con velocità di regime di 100 rpm

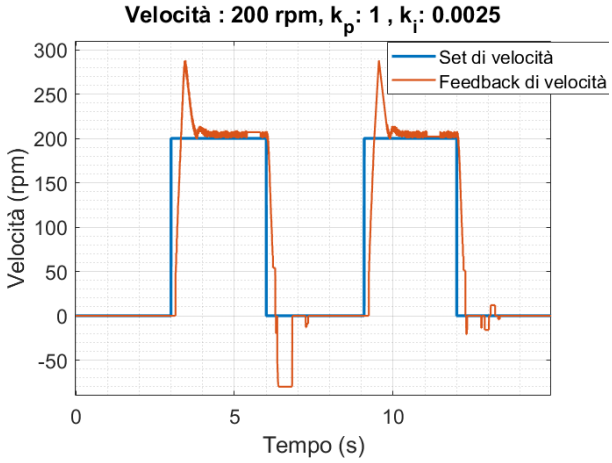


Figura 3.20: Prova comando gradino con velocità di regime di 200 rpm

3.5 Caratterizzazione del motore

A seguito dello studio del funzionamento del driver e del motore in controllo velocità, descritto nel paragrafo precedente, è stato condotta l'analisi del funzionamento in controllo coppia. In questo caso il motore viene comandato per mezzo della variazione di un solo parametro, ossia la corrente di quadratura i_q. Come descritto nel paragrafo 3.1.3 questa corrente è responsabile della variazione di flusso magnetico all'interno del motore, determinando il valore di coppia esercitato. La corrente di quadratura non fornisce, però, alcuna informazione quantitativa sulla coppia esercitata dal motore, per cui si rende necessario ricavare una relazione tra le due per la caratterizzazione del motore. A tal fine è stata realizzata un'interfaccia con la funzione di collegare una cella di carico che blocchi la rotazione e misuri la forza esercitata dal motore. In figura 3.21 viene mostrato lo schema adottato per le prove di caratterizzazione, dal quale è possibile osservare l'interfaccia realizzata. Con **d** si indica la distanza tra l'asse di rotazione della ruota e la mezzeria delle viti di collegamento della cella di carico (d = 0,083 m).

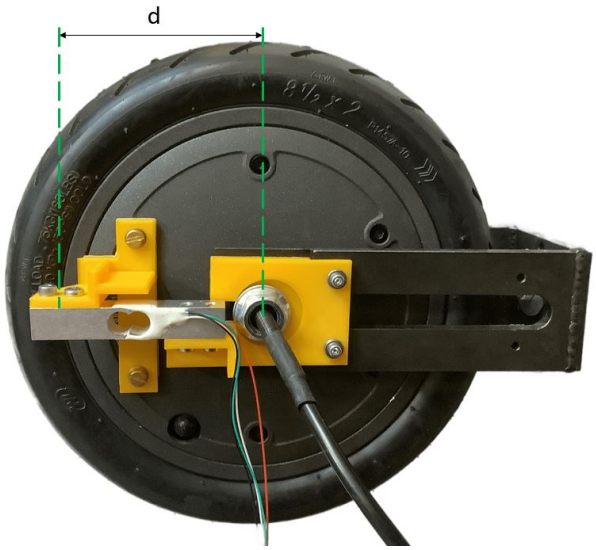
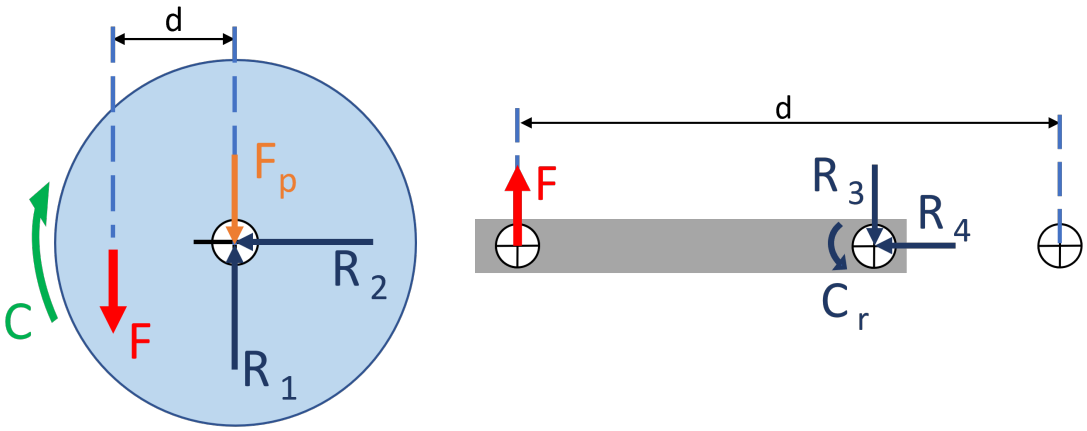


Figura 3.21: Schema prova di caratterizzazione motore



(a) *Diagramma di corpo libero ruota* **(b)** *Diagramma di corpo libero cella di carico*

Figura 3.22: Diagramma di corpo libero prova di caratterizzazione del motore

In figura 3.22 vengono rappresentati i diagrammi di corpo libero per la ruota (fig. 3.22a) e per la cella di carico (fig. 3.22b). Dove con **C** si indica la coppia esercitata dal motore, con **F** la forza misurata dalla cella di carico, con $\mathbf{F_p}$ la forza peso della ruota e con $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, $\mathbf{R_4}$ e $\mathbf{C_r}$ le reazioni vincolari.
Prendendo come riferimento la configurazione di figura 3.21 la cella di carico è stata calibrata attraverso la misurazione di un peso noto appendendolo nella mezzeria delle viti di collegamento. La coppia motrice è facilmente ricavabile conoscendo la forza **F** :

$$C = F \cdot d \tag{3.2}$$

A questo punto sono state effettuate le prove di caratterizzazione utilizzando il sistema rappresentato in figura 3.23.
Per completezza in tabella 3.4 viene riportato l'ordine in cui sono state effettuate le prove e la denominazione delle stesse nel paragrafo. Sono state effettuate due tipi di prove: la prima facendo crescere gradualmente il valore della corrente; la seconda dando un primo valore di corrente elevato per poi farla decrescere gradualmente.

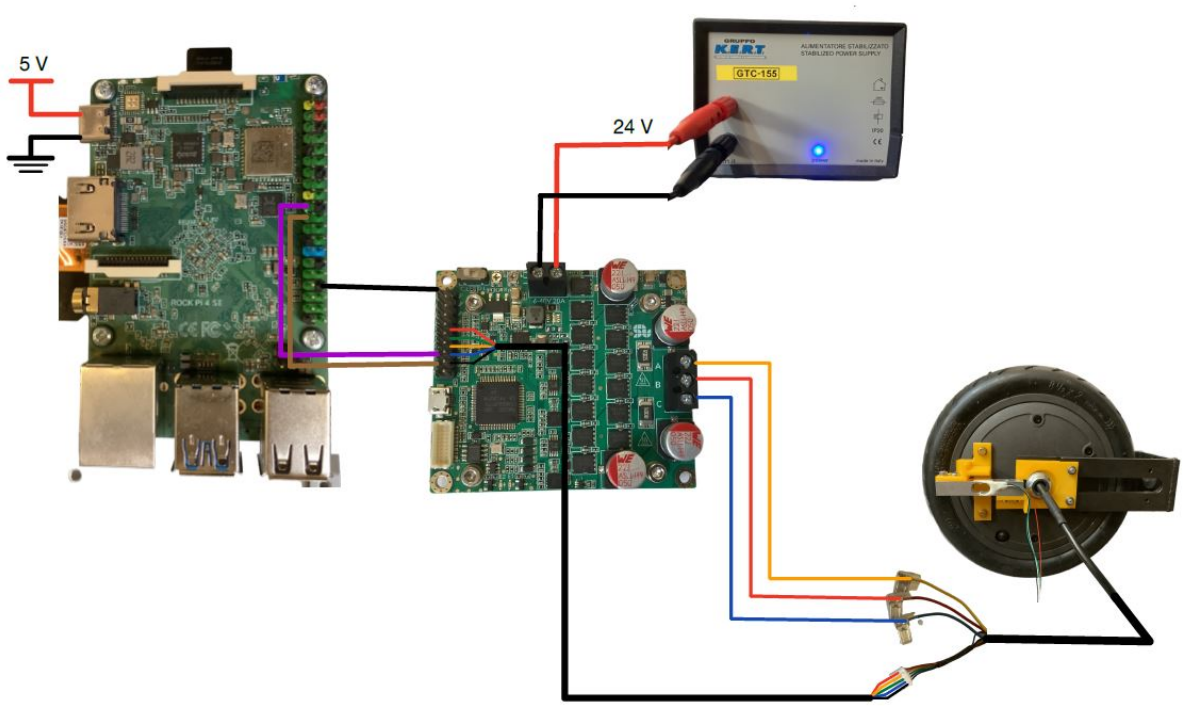
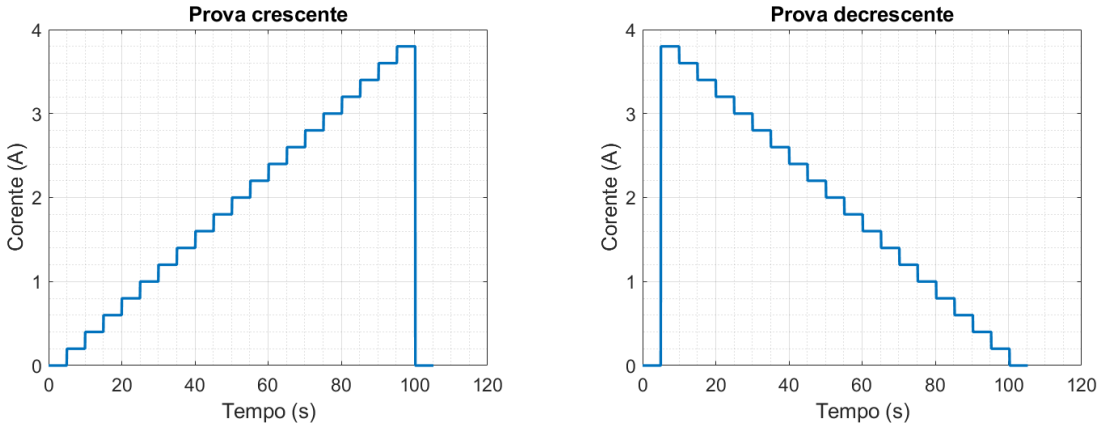


Figura 3.23: Banco prova caratterizzazione motore

PROVA	Denominazione
Taratura	
Prova gradino crescente 1	S1
Prova gradino crescente 2	S2
Prova gradino crescente 3	S3
Misurazione del peso campione	
Prova gradino crescente 4	S4
Prova gradino crescente 5	S5
Misurazione del peso campione	
Prova gradino decrescente 1	D1
Prova gradino decrescente 2	D2
Prova gradino decrescente 3	D3
Misurazione del peso campione	
Prova gradino decrescente 4	D4
Prova gradino decrescente 5	D5
Misurazione del peso campione	

Tabella 3.4: Prove di caratterizzazione del motore

In figura 3.24 vengono riportati i set di corrente di quadratura (i_q) per i due tipi di prova. Nel caso della prova crescente (fig. 3.24a) si può osservare che sono stati dati come comando dei gradini di corrente aumentando di 0,2 A ogni 5 secondi, fino ad un valore di 3,8 A; nel caso di prova decrescente (fig. 3.24b), invece, si parte da un valore di 3,8 A e si diminuisce il set di corrente di 0,2 A ogni 5 secondi.
Per ogni prova è stata misurata la forza esercitata sulla cella di carico ed è stata calcolata la coppia corrispondente. In seguito, sincronizzando nel tempo i due segnali, è stato possibile rappresentare il loro andamento. A titolo di esempio in figura 3.25 vengono mostrati gli andamenti risultanti dalle prove S3 e D3.

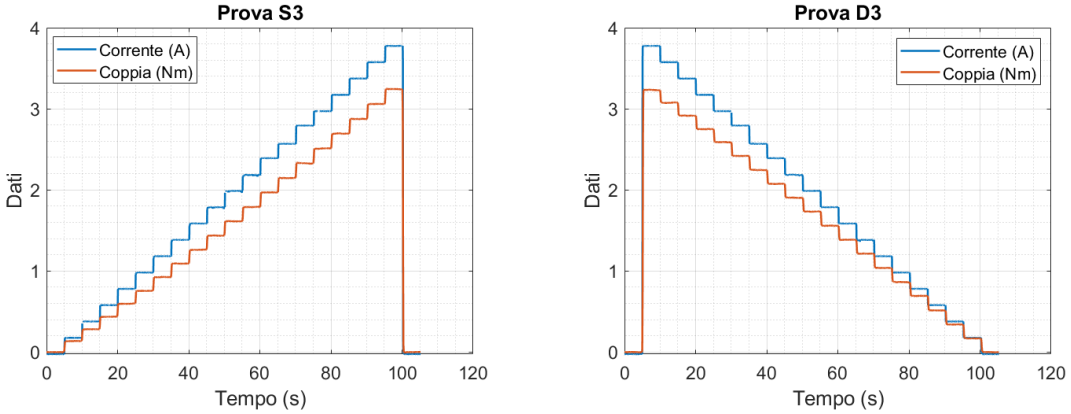


(a) *Test con comando crescente* **(b)** *Test con comando decrescente*

Figura 3.24: Set di corrente utilizzato per le prove di caratterizzazione

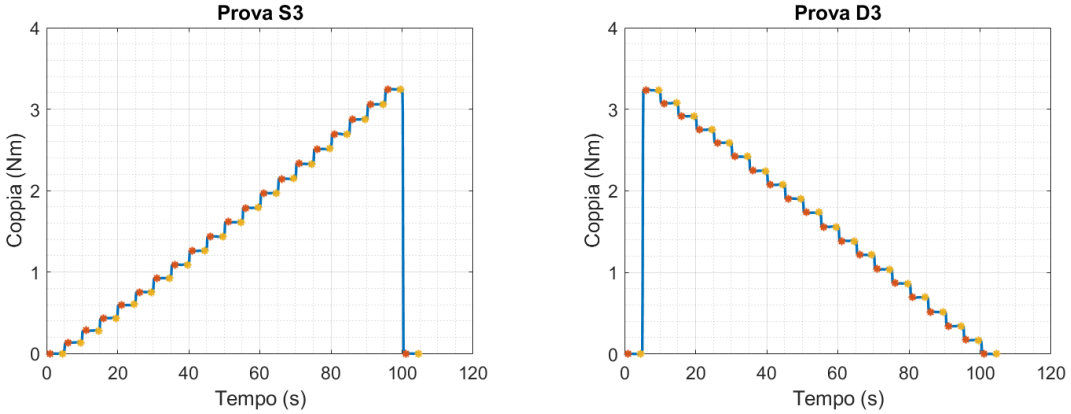
Per poter procedere a un'analisi quantitativa, in ogni intervallo in cui il valore della coppia resta costante, è stato possibile calcolarne il valor medio. In figura 3.26 sono stati messi in evidenza gli intervalli su cui questi valori sono stati calcolati. Inoltre, tramite il driver è possibile acquisire il valore di feedback della corrente di quadratura, da cui è possibile calcolarne il valor medio negli

stessi intervalli di tempo utilizzati per il calcolo del valor medio della coppia. In figura 3.27 sono illustrati gli intervalli utilizzati.



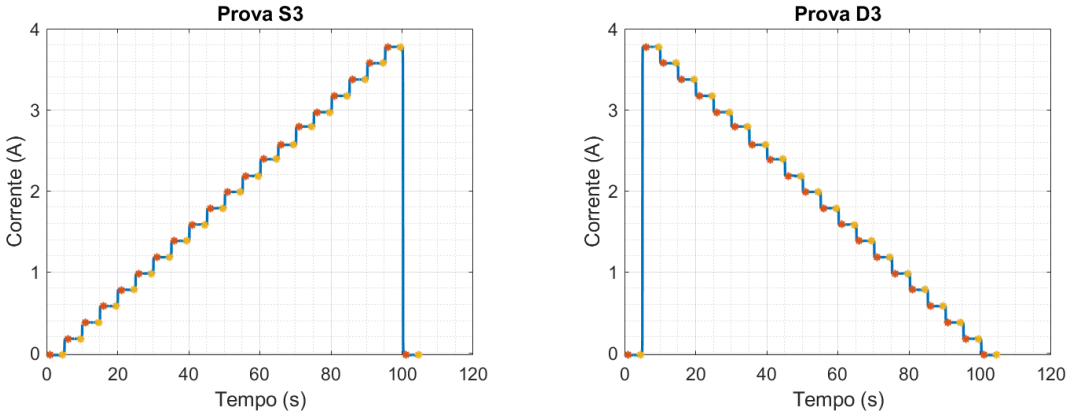
(a) *Test con comando crescente* **(b)** *Test con comando decrescente*

Figura 3.25: Confronto corrente di feedback e coppia erogata dal motore



(a) *Test con comando crescente* **(b)** *Test con comando decrescente*

Figura 3.26: Intervalli utilizzati per il calcolo del valor medio della coppia



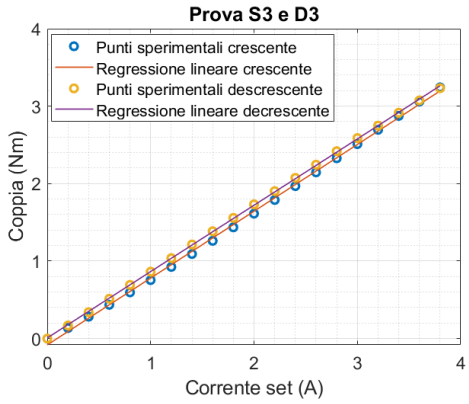
(a) *Test con comando crescente* **(b)** *Test con comando decrescente*

Figura 3.27: Intervalli utilizzati per il calcolo del valor medio della corrente

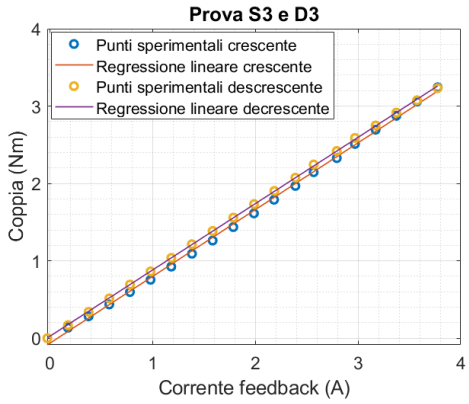
Successivamente, per ogni prova è stato possibile rappresentare l'andamento della coppia calcolata in funzione della corrente. In figura 3.28 sono indicati con un simbolo circolare i punti sperimentali ed è rappresentata con una retta la regressione lineare calcolata a partire da questi. In questo caso gli andamenti delle due prove considerate a titolo esemplificativo (S3 e D3) vengono mostrati nello stesso grafico per poterli confrontare. Sono stati riportati sia gli andamenti rispetto alla corrente impostata come set (fig. 3.28a), sia gli andamenti rispetto alla corrente di feedback (fig. 3.28b). Si può notare come la coppia abbia un andamento abbastanza lineare nel caso di prova decrescente, mentre nel caso di prova crescente presenta una lieve concavità.
Per ogni prova è stato possibile calcolare un coefficiente di coppia k_t relativo ad ogni intervallo come:

$$k_t = \frac{C}{i_q} \tag{3.3}$$

Mediando i valori ottenuti per ogni intervallo è possibile calcolare un valore del coefficiente di coppia relativo ad ogni prova, che altro non è che il coefficiente angolare della retta di regressione lineare presente in figura 3.28. Inoltre, per ogni prova è stato possibile calcolare due valori di coefficiente di coppia, uno calcolato tramite la corrente di quadratura data come set e l'altro calcolato utilizzando la corrente di quadratura di feedback.
I valori dei k_t medi di ogni prova sono indicati in figura 3.29, sia riferiti alla corrente di set (fig. 3.29a), sia rispetto alla corrente di feedback (fig. 3.29b).

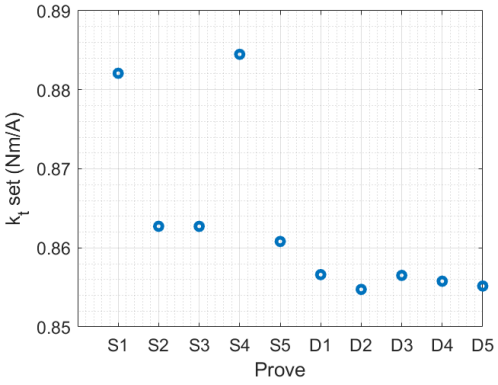


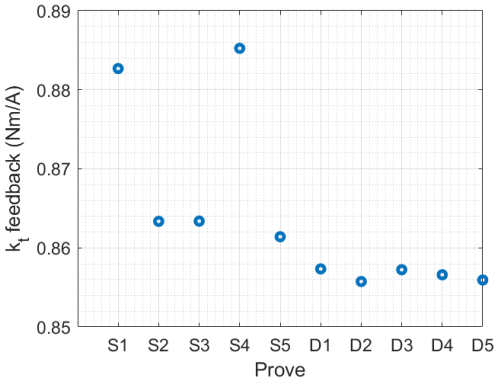
(a) *Regressione lineare rispetto alla corrente di set*



(b) *Regressione lineare rispetto alla corrente di set*

Figura 3.28: Regressione lineare della coppia rispetto alla corrente di set e feedback



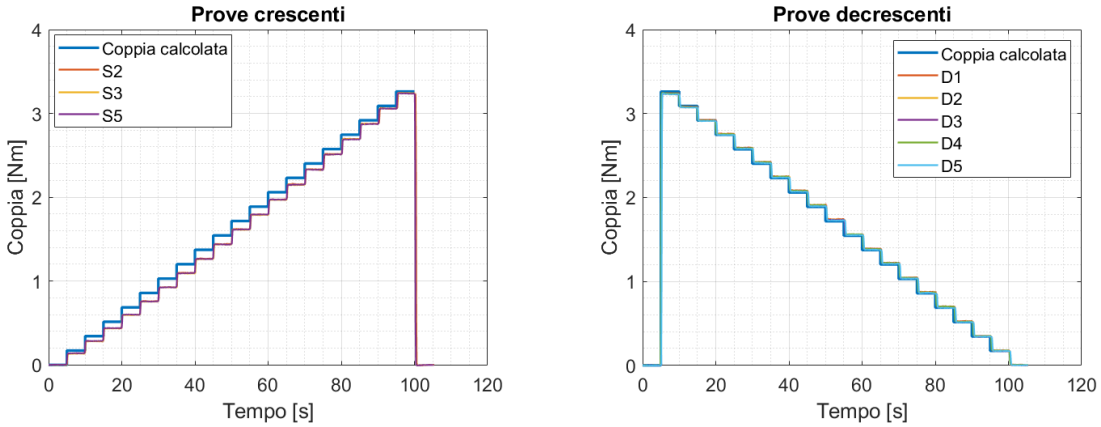


(a) *Coefficienti calcolati rispetto alla corrente di set* **(b)** *Coefficienti calcolati rispetto alla corrente di feedback*

Figura 3.29: Coefficienti di coppia calcolati per le diverse prove

È possibile notare dalle figure precedenti che i coefficienti ottenuti dalle prove S1 e S4 si discostano abbastanza da quelli ottenuti nelle altre prove. Per questo motivo è stato deciso di non considerare tali valori nei calcoli successivi in quanto potrebbero falsare i risultati.
A questo punto è stato calcolato un unico coefficiente di coppia per le prove di set e uno per le prove di feedback come media tra i coefficienti delle singole prove. Il k_t calcolato tramite la corrente di feedback è il coefficiente di coppia reale del motore. L'obbiettivo, però, è quello di comandare in coppia il motore e perciò è di maggiore interesse il k_t calcolato tramite la corrente di set.
Il coefficiente di coppia rispetto alla corrente di set ottenuto vale: $k_t = 0,858 \frac{Nm}{A}$.

In figura 3.30, per concludere, è stato riportato l'andamento della coppia calcolata utilizzando il valore del coefficiente di coppia ottenuto in precedenza. Nello specifico sia per i test con coppia crescente (fig. 3.30a), che per coppia decrescente (fig. 3.30b), oltre all'andamento della coppia calcolata, sono riportati, come confronto, gli andamenti delle prove sperimentali effettuate.



(a) *Test con comando crescente* **(b)** *Test con comando decrescente*

Figura 3.30: Confronto tra coppia calcolata e coppia sperimentale

Dai grafici si può notare come nelle prove in cui il valore della corrente viene aumentato gradualmente l'andamento della coppia calcolata in funzione del tempo si discosta di poco dagli andamenti rilevati nei test sperimentali. Nelle prove in cui il valore della corrente descresce gradualmente, al contrario, l'andamento della coppia calcolata nel tempo è praticamente del tutto sovrapponibile agli andamenti sperimentali risultati dai test eseguiti.

3.6 Test frenata e rigenerazione

L'ultimo test preliminare eseguito sul sistema consiste nella simulazione della frenata attiva del dispositivo. La messa in pratica di questo test prevede la realizzazione di un nuovo sistema in cui è presente l'utilizzo di due ruote, che per comodità saranno indicate come ruota 1 e ruota 2 come in figura 3.31. Nello specifico, per questa simulazione la ruota 1 deve essere trascinata in modo indipendente dalla corrente imposta al motore. Per fare ciò è stata utilizzata la seconda ruota motorizzata che viene posta a contatto con la ruota 1 per metterla in moto. La ruota 2 è controllata in velocità in modo da non essere influenzata dalla coppia frenante generata dalla ruota 1. Per questa prova è stata utilizzata la batteria del dispositivo, una batteria Panasonic con un voltaggio di 36 V e una capacità di 4,4 Ah. Inoltre, sul ramo positivo di alimentazione tra batteria e driver del motore è stato inserito un multimetro, con il quale è stato possibile leggere il valore della corrente che scorreva tra i due componenti e il verso della stessa. In figura 3.31 viene illustrato il sistema realizzato per il test.

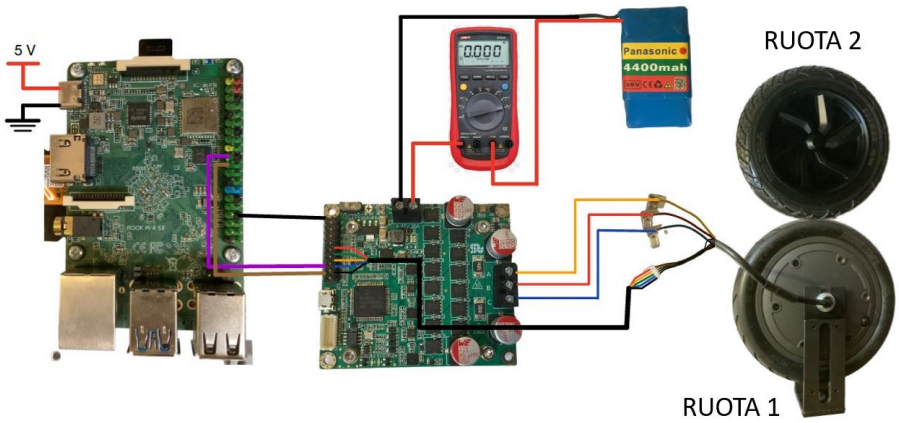


Figura 3.31: Schema di collegamento prove di frenata

Durante l'esecuzione della prova, veniva imposta, al motore della ruota 1, una coppia con verso opposto a quello di rotazione dello stesso. A questo punto era possibile osservare una diminuzione della velocità della ruota, mentre dal multimetro era possibile notare che per piccoli valori di coppia e di velocità di rotazione la corrente necessaria a frenare il sistema fosse uscente dalla batteria. Al contrario, aumentando uno dei due valori (coppia o velocità) la corrente in uscita dalla batteria inizia a diminuire fino a invertire il suo verso. In queste condizioni è possibile ricaricare la batteria.
L'importanza di queste prove deriva dal fatto che la carrozzina in alcune condizioni, come nel caso di discese, abbia la necessità di essere frenata. In questo senso diventa fondamentale il fatto che il driver permetta di fornire assistenza anche in situazioni di questo tipo. Inoltre, le prove sono state necessarie sia per testare la bidirezionalità della corrente all'interno del driver, sia per verificare che non ci fossero sovraccarichi di corrente sulla Rock Pi tali da disturbarne il funzionamento.

Capitolo 4

SOFTWARE

All'interno di questo capitolo verrà presentata l'architettura software utilizzata, fornendo una descrizione completa dei codici realizzati per il controllo del dispositivo.

4.1 Ambiente SOFTWARE

Sulla scheda Rock Pi 4 SE è stato installato Ubuntu, ossia un sistema operativo open source basato su Linux. Alcune delle sue principali caratteristiche includono un kernel Linux ottimizzato per hardware ARM, un'interfaccia utente personalizzabile con diversi ambienti desktop disponibili. Ubuntu su Rock Pi offre un sistema operativo versatile e stabile con una vasta gamma di software per vari scopi, dall'elaborazione dati all'uso come server domestico o media center.
Per la comunicazione con alcuni componenti hardware è necessario utilizzare determinati protocolli che non sono abilitati di default. Per abilitarne l'utilizzo sono state seguite le guide fornite da Rock Pi. In particolare è stato modificato il file di configurazione hardware *"hw_intfc.conf"*. Nello specifico, per questa applicazione è stato necessario abilitare la **UART 4** per poter comunicare con il driver SOLO, il **PWM 1** per poter utilizzare il servomotore e **I2C 6** per potersi interfacciare con la scheda MPU6050 montata sulla carrozzina.
Nel primo prototipo la frequenza di comando del sistema era data dalla somma dei contributi dei vari componenti; in questo modo era impossibile un controllo efficiente. Per poter ovviare a questo problema, è stato scelto di controllare il sistema meccatronico descritto nel capitolo precedente utilizzando il sistema operativo Robot Operating System (ROS). Infatti, ROS 2 permette la comunicazione parallelizzata dei vari componenti, in questo modo è possibile effettuare la comunicazione a frequenze differenti e specifiche per ogni componente. In questa applicazione, la lettura dai sensori viene effettuata a frequenze più elevate, in modo da avere delle informazioni più complete, il comando dei motori, invece, viene effettuato a frequenze più basse.

4.2 Descrizione ROS 2

Il Robot Operating System 2 (ROS 2)[24] è una piattaforma software open source sviluppata per supportare lo sviluppo di applicazioni robotiche. Una delle caratteristiche fondamentali di ROS 2 è la sua capacità di funzionare su una vasta gamma di piattaforme, tra cui Linux, Windows e MacOS. Una delle principali innovazioni di ROS 2 è l'architettura modulare, che sta a significare che il sistema è stato progettato in modo che i componenti software siano suddivisi in moduli separati che possono essere combinati in base alle esigenze specifiche del progetto. ROS 2 si è anche concentrato sulla gestione delle comunicazioni in tempo reale, aspetto cruciale in molte applicazioni robotiche. Il sistema offre un meccanismo di comunicazione avanzato e altamente efficiente che consente ai componenti software di scambiare dati e comandi con bassa latenza e alta affidabilità. Un'altra caratteristica chiave è il supporto per diversi linguaggi di programmazione, tra cui C++, Python e altri.
Sulla Rock Pi è stata installata la versione *"Foxy"* di ROS 2 [25], una versione Long Term Support (LTS) di ROS 2, importante per il fatto di essere una soluzione stabile e affidabile per progetti a lungo termine. Inoltre, *"Foxy"* è una delle prime versioni di ROS 2 a offrire un forte supporto per Python 3, il linguaggio scelto per lo sviluppo del progetto.
Dopo l'installazione del sistema operativo è necessaria la creazione di un workspace, che in ROS 2 rappresenta un ambiente di sviluppo isolato in cui è possibile organizzare i pacchetti e le dipendenze del proprio progetto. Le caratteristiche principali dei workspace ROS 2 sono l'isolamento, in quanto ogni workspace può avere le proprie dipendenze e versioni dei pacchetti senza interferire con altri workspace, la possibilità di organizzare i pacchetti in maniera gerarchica, la possibilità di creare variabili d'ambiente (in questo caso *"setup.bash"*) da eseguire per configurare l'ambiente di sviluppo e così includere le librerie e i pacchetti dal workspace corrente. Lo strumento principale di un workspace è il comando *Cmake Ordered Language CONsolidation (colcon)*, utilizzato per la compilazione e la gestione dei workspace di ROS 2. Esso semplifica il processo di compilazione dei pacchetti, gestisce le dipendenze e consente di mantenere un ambiente di sviluppo pulito.
Una volta creato il workspace di lavoro (in questo progetto denominato *add_on_ws*), è necessario inizializzarlo tramite il comando *colcon* e attivarlo.
A questo punto è possibile creare un nuovo package di lavoro, ossia un'unità di organizzazione logica per il codice che fornisce una struttura organizzativa chiara e semplificata per la gestione delle dipendenze e la distribuzione del software. Un package di ROS 2 contiene il codice sorgente, i file di configurazione, le risorse e le dipendenze necessarie. Ogni package di ROS 2 segue una struttura specifica che include directory per il codice sorgente (*"src"*), per il file di lancio (*"launch"*) e per i file di configurazione (*"config"*). In questo caso il package utilizzato è stato denominato *add_on_package*. A seguito della creazione del package è necessaria la sua configurazione inserendo dei codici sorgente e dei file di configurazione, aggiungendoli nelle directory corrispondenti. È necessaria anche la definizione delle dipendenze inserendo nel file *"package.xml"* l'elenco dei package dipendenti. Lo step successivo prevede che il workspace venga compilato in modo tale che il package risulti pronto per essere utilizzato.
Altro strumento presente in ROS 2 usato per la visualizzazione dei grafici di connessione dei nodi all'interno di un sistema ROS è l'*Rqt_graph*. Questo strumento è utile per comprendere come i nodi nel sistema comunichino tra loro, mostrando le connessioni tra i topic, i servizi e i parametri utilizzati dai nodi.

4.3 Descrizione dell'ambiente ROS 2

L'ambiente ROS realizzato per la nostra applicazione è rappresentato in figura 4.1: nello specifico in figura si possono osservare i nodi, rappresentati come ellissi, e i topic rappresentati come rettangoli. In ROS un nodo è un processo autonomo che esegue una specifica funzione all'interno di un sistema e contribuisce alla modularità, alla scalabilità e all'efficacia complessiva dell'applicazione robotica. Un topic, invece, è un canale di comunicazione asincrono utilizzato per scambiare dati tra i nodi del sistema. I topic sono uno dei principali metodi di comunicazione utilizzati in ROS per consentire la collaborazione tra i diversi nodi all'interno di un'applicazione robotica. Nella terminologia di ROS, il nodo che pubblica dati su un topic è chiamato *"publisher"*, mentre il nodo che riceve messaggi da un topic è chiamato *"subscriber"*. In un dato ambiente ROS un nodo può svolgere contemporaneamente sia la funzione di publisher che quella di subscriber.

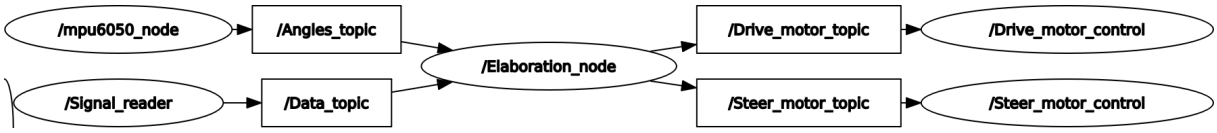


Figura 4.1: Rqt graph dell'ambiente ROS

Nell'ambiente ROS realizzato sono presenti 5 nodi:

- ***Mpu6050_node*** *: è il nodo di comunicazione con il sensore mpu6050;*
- ***Signal_reader*** *: è il nodo di comunicazione con le ruote della carrozzina;*
- ***Elaboration_node*** *: è il nodo che elabora i segnali in ingresso e calcola i segnali di riferimento per i due motori;*
- ***Drive_motor_control*** *: è il nodo di comando per il motore di spinta;*
- ***Steer_motor_control*** *: è il nodo di comando per il servomotore.*

E 4 topic:

- ***Angles_topic*** *: è il topic su cui vengono pubblicati gli angoli di inclinazione della carrozzina;*
- ***Data_topic*** *: è il topic su cui vengono pubblicate la forza esercitata su ogni pushrim, la rispettiva velocità angolare e l'inclinazione della strada;*
- ***Drive_motor_topic*** *: è il topic su cui viene pubblicata la coppia di riferimento per la spinta di assistenza;*
- ***Steer_motor_topic*** *: è il topic su cui viene pubblicato il riferimento per l'angolo di sterzata.*

Nel caso dell'applicazione reale è necessario che all'accensione tutto il sistema sia pronto per funzionare. Per consentire ciò, in primo lungo, è stato aggiunto come nuovo start-up program, un terminale eseguito come amministratore. Successivamente è stato modificato il file *".bashrc"* all'interno del quale vengono indicati i comandi da eseguire ogni qual volta si apre un nuovo terminale. Nello specifico, in questo file sono state aggiunte le righe di codice per eseguire i file di configurazione per l'ambiente ROS e per il workspace. In più è stata inserita l'indicazione per l'esecuzione del launch file. In generale, un launch file serve per eseguire i nodi desiderati di un workspace ROS. In questo caso i nodi vengono eseguiti nell'ordine prestabilito. Il codice del launch file è allegato in appendice B.1.
Si passa ora ad una descrizione completa di ogni nodo.

4.3.1 Nodo mpu6050

Il nodo *"Mpu6050_node"*, come detto in precedenza, è il nodo di comunicazione con il sensore IMU montato sul telaio della MWC. Nel codice, per prima cosa, bisogna impostare l'indirizzo della porta I2C a cui è collegato il modulo MPU 6050. Il nodo funziona ad una frequenza di 100 Hz e inizialmente legge i dati dall'accelerometro e del giroscopio, dai quali, tramite tecniche di sensor fusion, è possibile stimare gli angoli di orientazione *yaw* (ϕ), *pitch* (θ) e *roll* (ψ). In figura 4.2 è rappresentata la convenzione a cui fanno riferimento gli angoli precedentemente citati.

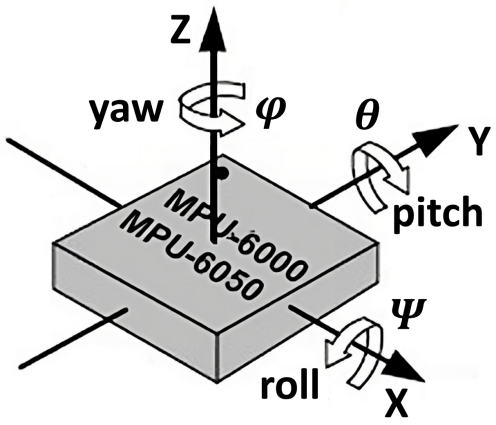


Figura 4.2: Rappresentazione angoli di orientazione

Chiamando rispettivamente a_x, a_y, a_z i dati dell'accelerometro e con $gyro_x$, $gyro_y$, $gyro_z$ i dati del giroscopio, viene illustrata la metodologia di calcolo utilizzata:

$$\begin{cases} \theta_{acc} = -\operatorname{atan2}(a_x\ ,\ \sqrt{a_y^2 + a_z^2}) \\ \psi_{acc} = -\operatorname{atan2}(a_y\ ,\ a_z) \end{cases} \tag{4.1}$$

dove θ_{acc} e ψ_{acc} sono rispettivamente gli angoli di beccheggio e rollio calcolati tramite i dati derivanti dall'accelerometro.

$$\begin{cases} \theta_g^i = \theta_g^{i-1} + gyro_y \cdot dt \\ \psi_g^i = \psi_g^{i-1} + gyro_x \cdot dt \end{cases} \tag{4.2}$$

Utilizzando l'equazione 4.2 è possibile calcolare i valori degli angoli di beccheggio e rollio utilizzando i dati misurati dal giroscopio. L'intervallo di tempo dt utilizzato è pari all'intervallo di campionamento, ovvero $dt = 0,01$ s.

$$\begin{cases} \theta = \alpha \cdot \theta_g + (1 - \alpha) \cdot \theta_{acc} \\ \psi = \alpha \cdot \psi_g + (1 - \alpha) \cdot \psi_{acc} \end{cases} \tag{4.3}$$

L'equazione 4.3 rappresenta l'applicazione del filtro complementare ai valori degli angoli di beccheggio e rollio; il valore di α indica il peso del giroscopio nel filtro complementare, in questo caso α = 0,98.

Nell'eq. 4.4 viene riportato per completezza il calcolo dell'angolo di imbardata, ma bisogna precisare che questo è un dato instabile e affetto da errore, per cui non viene usato per il controllo del sistema, ma solo come indicazione di variazione.

$$\phi = gyro_z \cdot dt \tag{4.4}$$

Dopo aver calcolato gli angoli ϕ, θ e ψ in gradi, ogni 0,01 s viene pubblicato un messaggio contenente il loro valore sul topic: *"Angles_topic"*. Il codice completo è riportato in appendice B.2.

4.3.2 Nodo Signal_reader

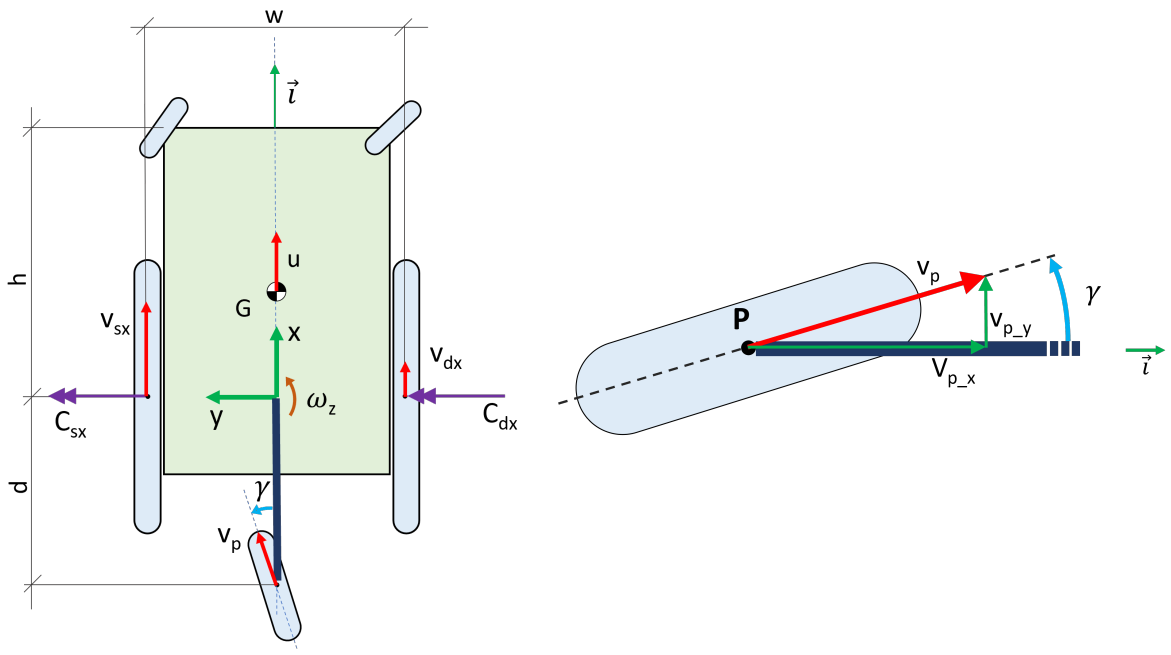
Il nodo *Signal_reader* è dedicato alla comunicazione con le schede montate sulle ruote. In particolare, sulle schede ESP32 montate su ogni ruota è in esecuzione un codice che legge i valori di velocità angolare del sensore IMU e i valori di forza dalle celle di carico. Una peculiarità delle schede ESP32 è quella di disporre di connettività bluetooth. Ciò è di notevole rilevanza in quanto la rotazione della ruota rende poco agevole sviluppare un collegamento via cavi. Scegliere quindi una soluzione wireless semplifica notevolmente la realizzazione della soluzione costruttiva e il montaggio delle ruote sulla MWC. La comunicazione tramite protocollo BLE avviene tra due dispositivi definiti rispettivamente *server* e *client*. Il dispositivo *server* espone i suoi dati attraverso una gerarchia di *servizi* e *caratteristiche*. Il *servizio* è un insieme di *caratteristiche* correlate che rappresentano una funzionalità specifica del dispositivo periferico. Entrambe le schede ESP32, presenti sulle ruote, sono state programmate per funzionare da server. Ciascun *server* definisce un *servizio* al cui interno è definita una *caratteristica*. Ogni *caratteristica* è definita come una variabile stringa contenente l'informazione di forza e velocità angolare convertiti in bit. Ad ogni ciclo il *server* aggiorna il valore della *caratteristica* con i nuovi dati letti dai sensori e invia una notifica al *client*. Il *client* nella comunicazione bluetooth è rappresentato dal microprocessore centrale, il quale legge il valore delle *caratteristiche* ogni volta che riceve una notifica dai *server*. Durante la fase di inizializzazione del nodo *Signal_reader*, avviene la connessione bluetooth del *client* ai *server* identificabili tramite il device address. Contemporaneamente, ad una frequenza di circa 100 Hz, vengono pubblicati sul *"Data_topic"* le ultime misure della ruota destra e sinistra. Il codice completo viene allegato in appendice B.3.

4.3.3 Nodo Elaboration

Questo è il nodo principale del sistema di assistenza. In questo nodo viene letto il valore di inclinazione della MWC, i valori di velocità angolare delle ruote e le coppie esercitate dall'utente. Questi dati sono elaborati per calcolare i valori di riferimento di coppia e angolo per il comando dei motori.

Calcolo angolo di sterzata

Il dispositivo di assistenza deve fornire, non solo una spinta longitudinale, ma anche un momento angolare per agevolare l'utente nell'esecuzione di curve strette e manovre angolari. Tuttavia, se l'angolo di sterzata non è coerente con la cinematica della carrozzina, si verifica uno slittamento del punto di contatto della ruota motrice con il terreno. La forza di attrito generata dallo scivolamento dà all'utente una sensazione di resistenza, annullando l'effetto di assistenza. Pertanto, la direzione della ruota di assistenza deve essere tale da evitare condizioni di strisciamento, migliorando l'assistenza del dispositivo. A partire dalla condizione cinematica iniziale e dai valori misurati dalle ruote, tramite un modello dinamico della carrozzina, è possibile stimare la condizione cinematica in un istante di tempo successivo. In particolare è stata definita una funzione che riceve in input la velocità angolare delle ruote e le forze esercitate dall'utente, e fornisce in output i valori di velocità lineare (u) e di velocità angolare attorno ad un asse verticale (ω_z) della carrozzina. In [26] è definito il modello implementato nella funzione. A partire dallo stato cinematico stimato in un istante di tempo successivo, risolvendo la cinematica del sistema, è possibile valutare l'angolo γ di rotazione della ruota. In figura 4.3 è riportata una rappresentazione schematica di carrozzina e dispositivo in una data configurazione, nello specifico in figura 4.3b è rappresentato un dettaglio della ruota del dispositivo.



(a) *Rappresentazione del modello completo* **(b)** *Dettaglio della ruota del dispositivo*

Figura 4.3: Rappresentazione del modello utilizzato per il calcolo dell'angolo γ

Di seguito viene riportato il calcolo dell'angolo γ:

$$\begin{cases} v_{p,x} = u \\ v_{p,y} = -d \cdot \omega_z \end{cases} \tag{4.5}$$

$$\gamma = \operatorname{atan} 2(v_{p,y}, v_{p,x}) \tag{4.6}$$

Dove $v_{p,x}$ e $v_{p,y}$ sono le velocità del punto di contatto della ruota del dispositivo nelle direzioni del sistema di riferimento della carrozzina, u è la velocità lungo la direzione x del baricentro della carrozzina, ω_z è la velocità angolare della carrozzina rispetto all'asse verticale, d è la distanza tra il punto di contatto della ruota del dispositivo e il baricentro della carrozzina e nello specifico d vale 0,3 m.
Nel codice è possibile cambiare l'istante di tempo in corrispondenza del quale stimare lo stato cinematico della MWC. Per l'applicazione è stato scelto di stimare la condizione cinematica della carrozzina dopo 0,2 s. È da notare che l'intervallo di tempo che intercorre tra il segnale di set e l'effettivo raggiungimento della condizione stimata è minore di quello scelto, in quanto bisogna tenere in considerazione il ritardo risultante nel segnale di comando dovuto ai tempi di calcolo.
Una volta calcolato, il valore di γ viene pubblicato sul topic: "Steer_motor_topic".

Calcolo coppia di assistenza

La legge di controllo implementata in questo codice tiene conto della forza esercitata dall'utente sui pushrim, dell'inclinazione del terreno e della velocità di marcia. Di conseguenza, il dispositivo è in grado di fornire un'assistenza più adeguata alle esigenze dell'utente e di adattarsi continuamente alle condizioni esterne [27]. La legge di controllo viene riportata nell'equazione 4.7.

$$C_{add-on}(t, v, \alpha) = (C_{spinta}(t) \cdot (a_1 - \frac{a_1 \delta}{2\delta_{max}}) + C_{spinta}(t-\delta)(\frac{a_1 \delta}{2\delta_{max}})) \cdot (1 - b_1 \alpha) + M_{tot} g \sin(\alpha) r \tag{4.7}$$

$$C_{spinta} = \frac{(F_{sx} + F_{dx}) \cdot R}{2} \tag{4.8}$$

$$\delta = \max(0; \min(d_1 v - d_2; \delta_{max})) \tag{4.9}$$

Parametri	a_1	$b_1(rad^{-1})$	$d_1(s^2 m^{-1})$	$d_2(s)$	$\delta_{max}(s)$	$r(m)$	$R(m)$
	0,75	6,02	0,67	0,17	0,25	0,11	0,26

Tabella 4.1: Parametri della legge di controllo

Dove:

- ***C_{add-on}*** *è la coppia di assistenza fornita dal dispositivo;*
- ***C_{spinta}*** *è il valor medio della coppia impartita dall'utente sui pushrim, calcolata come descritto nell'equazione 4.8;*
- ***F_{sx}*** *e* ***F_{dx}*** *sono le coppie misurate dalle celle di carico;*
- ***R*** *indica il raggio dei pushrim montati sulle ruote della MWC;*

- ***r*** *indica il raggio della ruota del dispositivo;*
- δ *è un fattore di ritardo descritto nell'equazione 4.9;*
- ***v*** *è la velocità della carrozzina;*
- α *è la pendenza misurata dall'IMU;*
- M_{tot} *è la massa totale;*
- a_1, b_1,d_1, d_2 *e* δ_{max} *sono i coefficienti definiti in tabella 4.1.*

La coppia totale $C_{add\text{-}on}$ è composta da tre contributi principali: proporzionale, ritardo e compensazione della gravità. La componente proporzionale rende il dispositivo reattivo alle rapide variazioni della coppia in ingresso ed è l'unico contributo a bassa velocità (v < 0,25 m/s). La componente di ritardo evita brusche variazioni della velocità, poiché garantisce una maggior durata della coppia di assistenza rispetto alla spinta impartita dall'utente. L'introduzione del contributo di ritardo rende migliore l'assistenza a velocità più elevate in condizioni di regime. La componente gravitazionale viene introdotta per compensare la gravità quando la carrozzina si trova su una rampa, impedendone l'arretramento e riducendo lo sforzo dell'utente in queste condizioni.
Il valore della $C_{add\text{-}on}$ viene pubblicato sul topic: *"Drive_motor_topic"*.
Il codice completo è allegato in appendice B.4.

4.3.4 Nodo Steer_motor

Il nodo *Steer_motor* ha la finalità di comandare il servomotore utilizzato per la sterzata della motoruota. In particolare, il segnale di comando per questo motore è di tipo PWM. Innanzitutto è necessario inizializzare la comunicazione con il motore per mezzo dei PIN che supportano la comunicazione PWM; per il servomotore è stato utilizzato il PIN 13 del microprocessore. È stata creata una variabile per il motore utilizzando la libreria di python *"mraa"* [28]. Sono stati impostati i parametri di controllo per il servomotore e sperimentalmente è stato individuato l'intervallo di comando che permette la variazione dell'angolo di 270° (0,06 - 0,186). Per trasmettere la rotazione sono state utilizzate due ruote dentate, come descritto nel paragrafo 3.5. Al fine di garantire il corretto angolo di sterzata della ruota è necessario tenere in considerazione il valore del rapporto di trasmissione tra le due ruote dentate, in modo da impostare il corretto angolo al servomotore. Si indica con γ l'angolo di rotazione della ruota del dispositivo, con θ l'angolo di rotazione del servomotore e con i il rapporto di trasmissione. Vale la relazione 4.10:

$$\theta = i \cdot \gamma \qquad con \qquad i = \frac{z_{ruota}}{z_{servo}} \tag{4.10}$$

Dove z_{ruota} e z_{servo} sono rispettivamente il numero di denti della ruota dentata collegata all'albero verticale del dispositivo e il numero di denti della ruota dentata collegata al servomotore.
Il calcolo del valore di set da assegnare al servomotore è mostrato nell'equazione 4.11, dove con x si indica il set, con x_{min} e x_{max} i limiti minimo e massimo, con θ_{min} e θ_{max} i valori limiti di sterzata ($\pm 135°$).

$$x = \frac{x_{min} - x_{max}}{\theta_{min} - \theta_{max}} \cdot (\theta - \theta_{max}) + x_{max} \tag{4.11}$$

Il codice completo è allegato in appendice B.5.

4.3.5 Nodo Drive_motor

L'ultimo nodo da descrivere è quello che serve a comandare il motore di spinta. Per fare ciò si utilizza la libreria SoloPy [29] messa a disposizione per il driver utilizzato. Nella fase di inizializzazione è innanzitutto necessario specificare quale tipo di comunicazione e baud rate si vuole utilizzare. Per la comunicazione tra driver e microprocessore è stata utilizzata la porta UART4 con un baud rate di 115200 bit/s. Inoltre, bisogna inizializzare i dati del motore inserendo il numero di poli (27), impostare una velocità massima e una corrente massima. Successivamente, per poter gestire la frequenza di comando dal codice ROS è stato necessario modificare il file della libreria, diminuendo i tempi di attesa imposti a seguito dell'esecuzione di ogni comando e in caso di disconnessione tra driver e microprocessore. La lettura della coppia di riferimento viene eseguita dal topic *"Drive_motor_topic"* e nel codice vengono effettuate due operazioni: imporre la direzione oraria nel caso in cui la coppia abbia un valore positivo e antioraria nel caso in cui sia negativo; calcolare il set da fornire al motore come segue:

$$i_q = \frac{C}{k_t} \tag{4.12}$$

dove i_q è il valore della corrente di quadratura che viene inserito come set per il motore.
Il codice completo è allegato in appendice B.6.

Capitolo 5

CONCLUSIONI

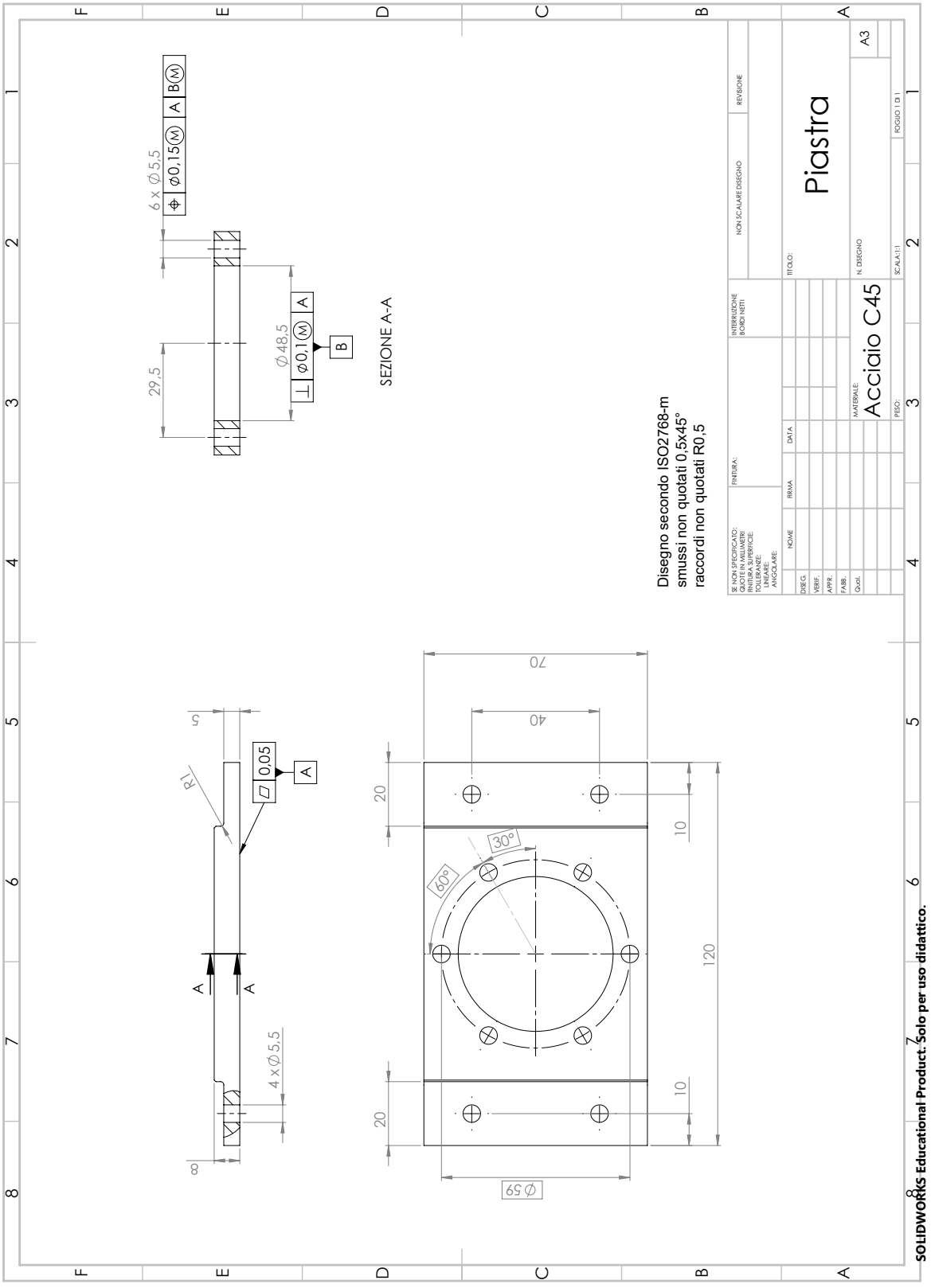
La tesi inizialmente presenta un'analisi sui dispositivi di assistenza alla propulsione di sedie a rotelle manuali attualmente disponibili sul mercato, identificandone sia i punti di forza che le limitazioni. Partendo da un dispositivo precedentemente realizzato, sono state proposte modifiche e revisioni mirate all'ottimizzazione dell'architettura complessiva. Queste revisioni architetturali sono state progettate per migliorare diversi aspetti del funzionamento del dispositivo.
Le modifiche strutturali apportate al dispositivo sono state valutate attraverso simulazioni FEM della struttura, che hanno rivelato una significativa riduzione della deformazione, pari a circa il 95%. Inoltre, per evitare infiltrazioni di polveri e proteggere l'hardware elettronico, sono stati progettati carter di copertura e protezione in materiale plastico, prevedendo la realizzazione con tecniche di fabbricazione additiva.
Alcuni componenti elettronici sono stati sostituiti per migliorare il controllo del dispositivo, in particolare il microprocessore e il driver del motore di spinta. L'implementazione di un controllo di coppia attraverso la modifica del driver del motore ha consentito maggiore versatilità e controllo del dispositivo. Sono stati anche condotti test per caratterizzare il motore al fine di identificare il coefficiente di coppia, un parametro essenziale per il controllo del motore. Inoltre, è stata testata la funzionalità di frenata rigenerativa, in questo modo è stato possibile integrare un controllo che fornisse assistenza anche durante la frenata.
La sostituzione del microprocessore ha permesso l'uso di ROS 2 per creare un'architettura software multisensore. In questo modo è stato possibile parallelizzare l'esecuzione dei codici di controllo, permettendo a ciascun componente hardware di funzionare alla frequenza ottimale, migliorando così il comportamento complessivo del sistema meccatronico.
Come sviluppi futuri, è possibile considerare l'implementazione di un metodo di controllo basato su algoritmi di model predictive control per il controllo di coppia, ottimizzando l'assistenza fornita dal dispositivo. Inoltre, la riprogettazione del sistema di aggancio del dispositivo alla carrozzina potrebbe migliorarne la facilità di installazione e la compatibilità con altri modelli di carrozzina. Queste prospettive offrono interessanti possibilità per l'evoluzione e il miglioramento del dispositivo di assistenza.

Appendice A

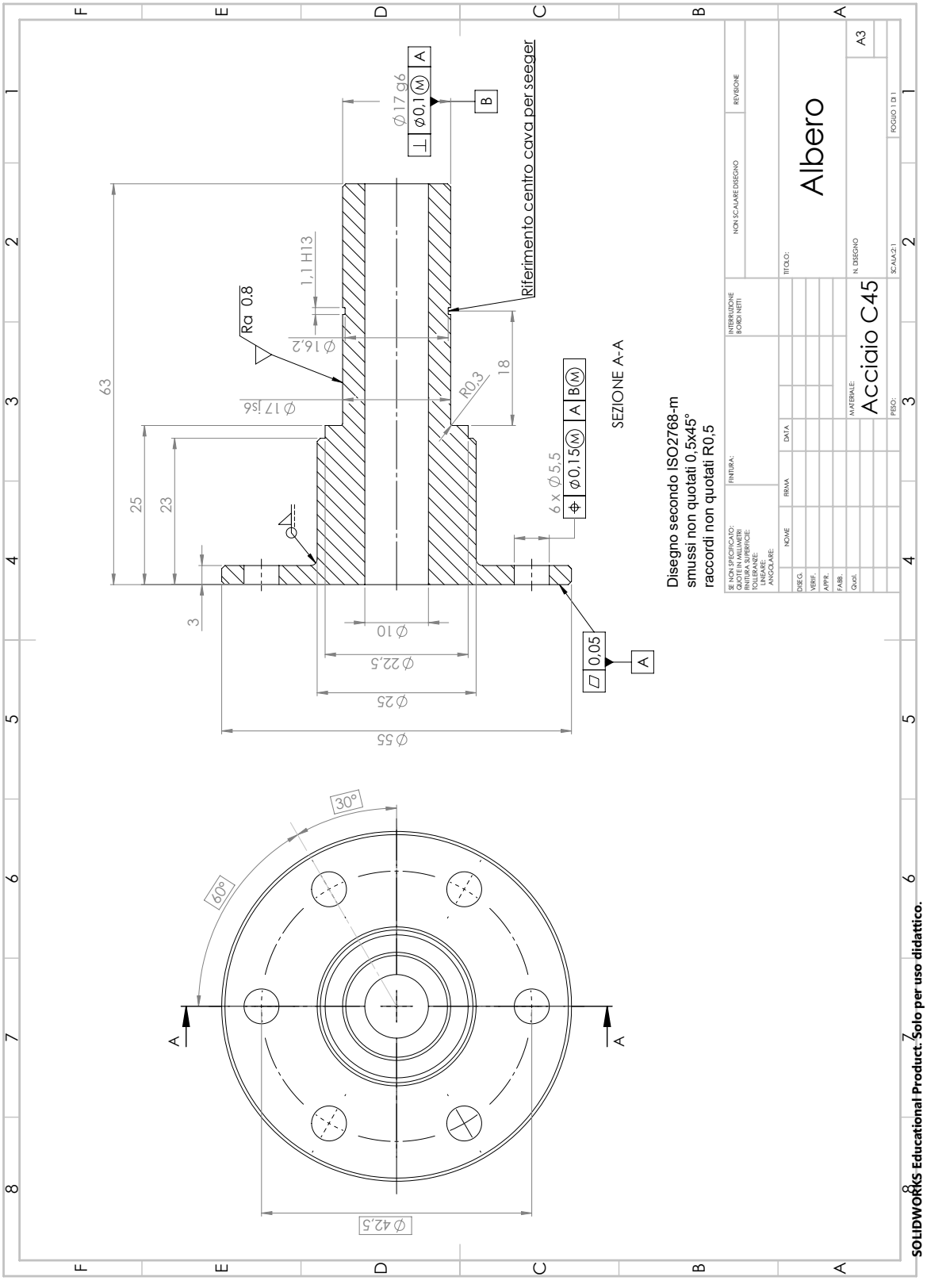
Tavole

In questa appendice vengono riportate le tavole realizzate secondo la normativa ISO2678-m.

A.1 Tavola piastra



A.2 Tavola albero



Appendice B

Codici

In seguito sono riportati il codice di lancio dell'architettura ROS 2 e i codici dei rispettivi nodi.

B.1 Launch file

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    ld=LaunchDescription()

    mpu = Node(
        package="add_on_package",
        executable="mpu",
    )

    ble = Node(
        package="add_on_package",
        executable="ble",
    )

    mpc = Node(
        package="add_on_package",
        executable="mpc"
    )

    steer = Node(
        package="add_on_package",
        executable="steer",
    )

    drive = Node(
        package="add_on_package",
        executable="drive",
    )

    ld.add_action(mpu)
    ld.add_action(ble)
    ld.add_action(mpc)
    ld.add_action(steer)
    ld.add_action(drive)

    return ld
```

B.2 MPU_6050_node

```python
import rclpy
from rclpy.node import Node
from std_msgs.msg import Float32MultiArray
from mpu6050 import mpu6050
import math

class MPU6050Node(Node):
    def __init__(self):
        super().__init__('mpu6050_node')
        print('inizio')
        self.sensor = mpu6050(0x68,6)  #I2C address of module MPU6050
        print('sensor ok')
        self.pitch=0
        self.roll=0
        self.publisher = self.create_publisher(Float32MultiArray, 'Angles_topic', 10)
        timer_period= 0.01      #changing the time sleep value changes the acquisition frequency
        self.timer = self.create_timer(timer_period, self.publish_angles)

    def publish_angles(self):
        yaw, pitch, roll = self.read_mpu6050_data()
        print(yaw,pitch,roll)
        angles_msg = Float32MultiArray()
        angles_msg.data=[yaw,pitch,roll]
        self.publisher.publish(angles_msg)

    def read_mpu6050_data(self):
        accel_data = self.sensor.get_accel_data()   #Reads accelerometer values
        gyro_data = self.sensor.get_gyro_data()     #Reads gyroscope values

        acc_x = accel_data['x']
        acc_y = accel_data['y']
        acc_z = accel_data['z']

        gyro_x = gyro_data['x']
        gyro_y = gyro_data['y']
        gyro_z = gyro_data['z']

        self.yaw, self.pitch, self.roll = self.calculate_angles(acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z)

        return self.yaw, self.pitch, self.roll

    def calculate_angles(self, acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z):
        # Constants for the complementary filter
        dt = 0.01  # (s)
        alpha = 0.98


        pitch = self.pitch
        roll = self.roll

        # pitch e roll calculation through acceleration

        pitch_acc = math.degrees(-math.atan2(acc_x , math.sqrt(acc_y^2+acc_z^2))
        roll_acc = math.degrees(-math.atan2(acc_y , acc_z))

        # pitch e roll calculation through gyroscope values
        pitch_g=pitch+gyro_y*dt
```

```
58          roll_g=roll+gyro_x*dt
59
60          # complementary filter
61          pitch = alpha * pitch_g + (1 - alpha) * pitch_acc
62          roll = alpha * roll_g + (1 - alpha) * roll_acc
63
64          yaw = gyro_z * dt
65
66          pitch_deg=math.degrees(pitch)
67          roll_deg=math.degrees(roll)
68          yaw_deg=math.degrees(yaw)
69
70          return yaw_deg, pitch_deg, roll_deg
71
72  def main(args=None):
73      rclpy.init(args=args)
74      mpu6050_node = MPU6050Node()
75      rclpy.spin(mpu6050_node)
76      rclpy.shutdown()
77
78  if __name__ == '__main__':
79      main()
```

B.3 Signal_reader

```
import rclpy
import random
import struct
import time
import numpy as np
from rclpy.node import Node
from std_msgs.msg import Float32,Float32MultiArray
from bluepy.btle import Peripheral
from bluepy import btle

value_ff=[0,0]
latest_data_sx=[0,0]
latest_data_dx=[0,0]

class MyDelegate_sx(btle.DefaultDelegate):
    def __init__(self):
        btle.DefaultDelegate.__init__(self)

    def handleNotification(self, cHandle, data):
        global latest_data_sx
        value = np.frombuffer(data,dtype=np.float32)
        for i in [0,1]:
            latest_data_sx[i]=float(value[i])

class MyDelegate_dx(btle.DefaultDelegate):
    def __init__(self):
        btle.DefaultDelegate.__init__(self)

    def handleNotification(self, cHandle, data):
        global latest_data_dx
        value = np.frombuffer(data,dtype=np.float32)
        for i in [0,1]:
            latest_data_dx[i]=float(value[i])

class BLECharacteristicReaderNode(Node):
    def __init__(self):
        super().__init__('Signal_reader')
        self.start=0
        self.publisher_ = self.create_publisher(Float32MultiArray, 'Data_topic',
 10)
        timer_period= 0.01  #changing the time sleep value changes the
acquisition frequency
        self.timer_ = self.create_timer(timer_period, self.
publish_characteristics)

        self.uuid_service_sx = "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
        self.uuid_service_dx = '6E400004-B5A3-F393-E0A9-E50E24DCCA9E'
        self.roll=float(0)
        self.uuid_sx = '6E400003-B5A3-F393-E0A9-E50E24DCCA9E'
        self.uuid_dx = '6E400006-B5A3-F393-E0A9-E50E24DCCA9E'

        self.device_address_sx = 'd4:d4:da:e4:3d:8a'
        self.device_address_dx = '08:3a:8d:2f:13:f6'

        setup_data=b"\x01\00"

        print('serching devices')
        self.sx = btle.Peripheral( self.device_address_sx )
        self.sx.setDelegate( MyDelegate_sx() )
        svc_sx = self.sx.getServiceByUUID(self.uuid_service_sx)
```

```
58          ch_sx = svc_sx.getCharacteristics(self.uuid_sx)[0]
59          self.sx.writeCharacteristic(ch_sx.valHandle+1,setup_data)
60          print('sx=ok')
61
62          self.dx = btle.Peripheral( self.device_address_dx )
63          self.dx.setDelegate( MyDelegate_dx() )
64          svc_dx = self.dx.getServiceByUUID(self.uuid_service_dx)
65          ch_dx = svc_dx.getCharacteristics(self.uuid_dx)[0]
66          self.dx.writeCharacteristic(ch_dx.valHandle+1,setup_data)
67          print('dx=ok')
68
69      def publish_characteristics(self):
70
71          if self.sx.waitForNotifications(0.1) & self.dx.waitForNotifications(0.1)
    :
72              msg = Float32MultiArray()
73              msg.data = [latest_data_sx[0],latest_data_sx[1],latest_data_dx[0],
    latest_data_dx[1]]
74              self.publisher_.publish(msg)
75
76
77  def main(args=None):
78      rclpy.init(args=args)
79      ble_characteristic_reader = BLECharacteristicReaderNode()
80      rclpy.spin(ble_characteristic_reader)
81      ble_characteristic_reader.destroy_node()
82      rclpy.shutdown()
83
84  if __name__ == '__main__':
85      main()
```

B.4 Elaboration_node

```
import rclpy
import math
import numpy as np
from rclpy.node import Node

from std_msgs.msg import Float32MultiArray


class ElaborationNode(Node):

    def __init__(self):
        super().__init__('Elaboration_node')
        self.ii=0
        self.a1=0.75
        self.b1=6.02
        self.d1=0.67
        self.d2=0.17
        self.delta_max=0.25
        self.r=0.11
        self.R=0.26
        self.M=90
        self.dim = 50
        self.coppia_sx_vec = np.zeros(self.dim)
        self.coppia_dx_vec = np.zeros(self.dim)
        self.subscription = self.create_subscription(Float32MultiArray,'Data_topic',self.listener_callback_data,10)
        self.subscription = self.create_subscription(Float32MultiArray,'Angles_topic',self.listener_callback_angles,10)
        self.publisher_Rotation= self.create_publisher(Float32MultiArray, 'Steer_motor_topic', 10)
        self.publisher_Drive = self.create_publisher(Float32MultiArray, 'Drive_motor_topic', 10)
        self.timer_period = 0.01  #changing the time sleep value changes the elaboration frequency
        self.timer = self.create_timer(self.timer_period, self.timer_callback)


    def listener_callback_data(self, msg):
        a=msg.data
        self.omg_sx=a[0]
        self.omg_dx=a[2]
        self.Csx=a[1]
        self.Cdx=a[3]

    def listener_callback_angles(self, msg):
        a=msg.data
        yaw=a[0]
        pitch=a[1]
        roll=a[2]
        self.incl=roll

    def timer_callback(self):

        msg = Float32MultiArray()
        msg.data = [self.mpc_rotation()]
        self.publisher_Rotation.publish(msg)
        self.get_logger().info('Publishing gamma rif: "%f"' % (msg.data[0]))
        msg_1 = Float32MultiArray()
        msg_1.data = [self.mpc_drive()]
        self.publisher_Drive.publish(msg_1)
```

```
56          self.get_logger().info('Publishing torque rif: "%f"' % (msg_1.data[0]))
57
58
59      def mpc_rotation(self):
60          omg_sx=self.omg_sx
61          omg_dx=self.omg_dx
62          Csx=self.Csx
63          Cdx=self.Cdx
64          n = 100  # num of iterations
65          dt = 0.002
66
67          # robot parameters
68          w = 0.52
69          d = 0.3
70          m = 96  # kg
71          Iz = 8.3
72          r_wheel = 0.3
73          xbc = 0.148
74          ybc = 0  # coordinates of COM
75          #c_viscous = 0
76          zetad0 = [(omg_dx + omg_sx) * r_wheel / 2, 0, (omg_dx - omg_sx) *
    r_wheel / w]
77          vp = [0, 0]
78          tau = [0, 0, 0]
79          D = [[m, 0, -ybc * m],
80               [0, m, xbc * m * 0],
81               [-ybc * m, xbc * m * 0, Iz + m * (xbc * xbc + ybc * ybc)]]
82          Dinv = np.linalg.inv(D)
83          zetad = zetad0[:]
84          zetad_old = zetad[:]
85          u, v, r = [0, 0, 0]
86          n_v = [0, 0, 0]
87          zetadd_temp = [0, 0, 0]
88          gamma = 0
89
90          Cdx = Cdx / 1000 * r_wheel * 9.81
91          Csx = Csx / 1000 * r_wheel * 9.81
92
93
94
95          for i in range(n):
96              u, v, r = zetad[:]
97
98              n_v[0] = -m * r * (v + xbc * r)
99              n_v[1] = m * r * (u - ybc * r) * 0  # zero based on differential
    wheels
100             n_v[2] = m * r * (xbc * u + ybc * v)
101
102             vp[0] = u
103             vp[1] = v - d * r
104
105             if abs(vp[0]) < 0.1:
106                 vp[0] = 0
107
108             if abs(vp[1]) < 0.1:
109                 vp[1] = 0
110
111             gamma = math.atan2(vp[1], vp[0])
112
113             tau[0] = (Cdx + Csx) / r_wheel
114             tau[1] = 0
115             tau[2] = -Csx / r_wheel * w / 2 + Cdx / r_wheel * w / 2
```

```
116
117             for j in range(3):
118                 zetadd_temp[j] = sum(Dinv[j][k] * (tau[k] - n_v[k]) for k in
    range(3))
119
120             zetadd = zetadd_temp[:]
121
122             zetad_old = zetad[:]
123             zetad[0] = zetad_old[0] + dt * zetadd[0]  # velocity update
124             zetad[1] = zetad_old[1] + dt * zetadd[1]  # velocity update
125             zetad[2] = zetad_old[2] + dt * zetadd[2]  # velocity update
126
127         if abs(gamma) > math.pi / 2:
128             gamma = gamma - math.pi * math.copysign(1, gamma)
129
130         if abs(vp[1]) < 0.01:
131             vp[1] = 0
132
133         return gamma
134
135
136     def mpc_drive(self):
137         omg_sx=self.omg_sx
138         omg_dx=self.omg_dx
139
140         for i in range(self.dim - 1):
141             self.coppia_sx_vec[i] = self.coppia_sx_vec[i + 1]
142             self.coppia_dx_vec[i] = self.coppia_dx_vec[i + 1]
143         self.coppia_sx_vec[self.dim - 1] = self.Csx
144         self.coppia_dx_vec[self.dim - 1] = self.Cdx
145
146         mean_omg=float((omg_sx + omg_dx)/2)
147
148         C_spinta_1=(self.Csx+self.Cdx)/1000 * self.R * 9.81/2
149
150         delta= max(0;min(self.d1*mean_omg-self.d2;self.delta_max))
151         j=int(round(delta/self.timer_period))
152         C_spinta_2=(self.coppia_sx_vec[self.dim-j-1]+self.coppia_dx_vec[self.dim
    -j-1])/1000 * self.R * 9.81/2
153
154         alpha=radians(self.incl)
155
156         c=(self.a1-delta)/(2*self.delta_max)
157
158         C_addon=(C_spinta_1*(self.a1-c)+C_spinta_2*c)*(1-self.b1*alpha)+self.M
    *9.81*math.sin(alpha)*self.r
159         C_addon=self.sat(C_addon, -4, 4)
160         return float(C_addon)
161
162
163
164     def sat(self, value, mini, maxi):
165         if value < mini:
166             return mini
167         elif value > maxi:
168             return maxi
169         else:
170             return value
171
172
173
174 def main(args=None):
```

```
175     rclpy.init(args=args)
176     elab_node = ElaborationNode()
177     rclpy.spin(elab_node)
178     elab_node.destroy_node()
179     rclpy.shutdown()
180
181
182 if __name__ == '__main__':
183     main()
```

B.5 Steer_motor_control

```
import rclpy
import time
import mraa
import numpy as np
from rclpy.node import Node

from std_msgs.msg import Float32MultiArray

class SteerSubscriber(Node):

    def __init__(self):
        super().__init__('Steer_motor_control')
        self.servo_pin = 13          #set the PWM PIN

        self.motor=mraa.Pwm(self.servo_pin)
        self.motor.period_us(20000)
        self.motor.enable(True)
        #setting
        self.angle=0
        self.x_min=0.06
        self.x_max=0.168
        self.angle_min=-215
        self.angle_max=270-215
        z_ruota=47       #number of device whhel gear teeth
        z_servo=31       #number of servomotor gear teeth
        self.i=z_ruota/z_servo

        self.subscription = self.create_subscription(Float32MultiArray,'Steer_motor_topic',self.listener_callback,10)
        timer_period= 0.1 #changing the time sleep value changes set frequency
        self.timer = self.create_timer(timer_period, self.timer_callback)


    def listener_callback(self, msg):
        self.a=msg.data
        self.angle=np.degrees(self.a[0])
        self.get_logger().info('I heard: "%f"' % (self.angle))




    def timer_callback(self):
        angle=self.angle*self.i
        x=((self.x_min-self.x_max)/(self.angle_min-self.angle_max))*(angle-self.angle_max)+self.x_max
        self.motor.write(x)

def main(args=None):
    rclpy.init(args=args)
    steer_motor = SteerSubscriber()
    rclpy.spin(steer_motor)
    steer_motor.destroy_node()
    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

B.6 Drive_motor_control

```
import rclpy
from rclpy.node import Node
import SoloPy as solo
import numpy as np
import matplotlib.pyplot as plt
import time

from std_msgs.msg import Float32MultiArray


class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('Drive_motor_control')

        self.kt=0.858       #set the torque coefficient value

        # instanciate a SOLO object:
        self.mySolo = solo.SoloMotorControllerUart("/dev/ttyS4", 0, solo.UART_BAUD_RATE.RATE_115200)    #use this for UART Communication
        #mySolo = solo.SoloMotorControllerUart("/dev/ttyACM0", 0, solo.UART_BAUD_RATE.RATE_937500)      #use this for USB Communication

        # Desired Switching or PWM Frequency at Output
        pwmFrequency = 20

        # Motor's Number of Poles
        numberOfPoles = 27

        # Current Limit of the Motor
        currentLimit = 4

        # Battery or Bus Voltage
        busVoltage = 0

        # Motor Torque feedback
        actualMotorTorque = 0

        # Motor speed feedback
        actualMotorSpeed = 0

        # Motor position feedback
        actualMotorPosition = 0

        # Motor speed limit
        speedLimit = 400

        # wait here till communication is established
        print("Trying to Connect To SOLO")
        communication_is_working = False
        while communication_is_working is False:
            time.sleep(1)
            communication_is_working, error = self.mySolo.communication_is_working()
        print("Communication Established succuessfully!")

        self.mySolo.overwrite_error_register()

        # Initial Configuration of the device and the Motor
        self.mySolo.set_output_pwm_frequency_khz(pwmFrequency)
```

```
58          self.mySolo.set_current_limit(currentLimit)
59          self.mySolo.set_motor_poles_counts(numberOfPoles)
60          self.mySolo.set_command_mode(solo.COMMAND_MODE.DIGITAL)
61          self.mySolo.set_motor_type(solo.MOTOR_TYPE.BLDC_PMSM)
62          self.mySolo.set_feedback_control_mode(solo.FEEDBACK_CONTROL_MODE.
    HALL_SENSORS)
63          self.mySolo.set_control_mode(solo.CONTROL_MODE.TORQUE_MODE)
64          self.mySolo.set_speed_limit(speedLimit)
65
66          # run the motor identification to Auto-tune the current controller gains
     Kp and Ki needed for Torque Loop
67          # run ID. always after selecting the Motor Type!
68          # ID. doesn't need to be called everytime, only one time after wiring up
     the Motor will be enough
69          # the ID. values will be remembered by SOLO after power recycling
70
71          # self.mySolo.motor_parameters_identification(solo.ACTION.START)
72          # print("Identifyng the Motor")
73          # # wait at least for 2sec till ID. is done
74          # time.sleep(2)
75
76          self.subscription = self.create_subscription(Float32MultiArray,'
    Drive_motor_topic',self.listener_callback,10)
77          timer_period= 0.1       #changing the time sleep value changes set
    frequency
78          self.timer = self.create_timer(timer_period, self.timer_callback)
79
80      def listener_callback(self, msg):
81          self.C=msg.data
82
83      def timer_callback(self):
84          if self.C[0]>0:
85              self.mySolo.set_motor_direction(solo.DIRECTION.CLOCKWISE)
86          elif self.C[0]<0:
87              self.mySolo.set_motor_direction(solo.DIRECTION.COUNTERCLOCKWISE)
88          set_current=abs(self.C[0]/self.kt)
89          self.mySolo.set_torque_reference_iq(set_current)
90
91  def main(args=None):
92      rclpy.init(args=args)
93
94      drive_motor = MinimalSubscriber()
95      drive_motor.start=time.time()
96      rclpy.spin(drive_motor)
97      drive_motor.destroy_node()
98      rclpy.shutdown()
99
100
101 if __name__ == '__main__':
102     main()
```

Bibliografia

[1] World Health Organization. Regional Office for South-East Asia. *Fact sheet on wheelchairs*. WHO Regional Office for South-East Asia, 2010.

[2] *Wheelchair Needs In The World*. URL: https://www.wheelchairfoundation.org/fth/analysis-of-wheelchair-need/.

[3] *Wheelchair provision guidelines*. World Health Organization, 2023. URL: https://www.who.int/publications-detail-redirect/9789240074521.

[4] Mahsa Khalili, Angela Eugenio, Allison Wood, Machiel Van der Loos, W. Ben Mortenson e Jaimie Borisoff. "Perceptions of power-assist devices: interviews with manual wheelchair users". In: *Disability and Rehabilitation: Assistive Technology* 18 (5 lug. 2023), pp. 693–703. ISSN: 1748-3107. DOI: 10.1080/17483107.2021.1906963.

[5] S. David Algood, Rory A. Cooper, Shirley G. Fitzgerald, Rosemarie Cooper e Michael L. Boninger. "Effect of a pushrim-activated power-assist wheelchair on the functional capabilities of persons with tetraplegia". In: *Archives of Physical Medicine and Rehabilitation* 86 (3 mar. 2005), pp. 380–386. ISSN: 00039993. DOI: 10.1016/j.apmr.2004.05.017.

[6] HK Lakomy, I Campbell e C Williams. "Treadmill performance and selected physiological characteristics of wheelchair athletes." In: *British journal of sports medicine* 21.3 (1987), pp. 130–133.

[7] ROGER M Glaser, MICHAEL N Sawka, LLOYD L Laubach e AGARAM G Suryaprasad. "Metabolic and cardiopulmonary responses to wheelchair and bicycle ergometry". In: *Journal of Applied Physiology* 46.6 (1979), pp. 1066–1070.

[8] MICHAEL N Sawka, ROGER M Glaser, STEPHEN W Wilde e THOMAS C von Luhrte. "Metabolic and circulatory responses to wheelchair and arm crank exercise". In: *Journal of applied physiology* 49.5 (1980), pp. 784–788.

[9] Rory A. Cooper, Shirley G. Fitzgerald, Michael L. Boninger, Karin Prins, Andrew J. Rentschler, Julianna Arva e Thomas J. O'Connor. "Evaluation of a pushrim-activated, power-assisted wheelchair". In: *Archives of Physical Medicine and Rehabilitation* 82 (5 mag. 2001), pp. 702–708. ISSN: 00039993. DOI: 10.1053/apmr.2001.20836.

[10] S. David Algood, Rory A. Cooper, Shirley G. Fitzgerald, Rosemarie Cooper e Michael L. Boninger. "Impact of a pushrim-activated power-assisted wheelchair on the metabolic demands, stroke frequency, and range of motion among subjects with tetraplegia". In: *Archives of Physical Medicine and Rehabilitation* 85 (11 nov. 2004), pp. 1865–1871. ISSN: 00039993. DOI: 10.1016/j.apmr.2004.04.043.

[11] Valerio Cornagliotto, Flaminia Perino, Laura Gastaldi e Stefano Pastorelli. "Evaluation on Implementing an Active Braking System in Wheelchair Rear-Mounted Power-Assisted Device". In: 2022, pp. 351–358. DOI: 10.1007/978-3-031-04870-8_41.

[12] Manveer Dhaliwal, Sara Janssen, Kylie Kuik e Edward Giesbrecht. “Choosing a Power Assist Device”. In: (gen. 2022). DOI: 10.13140/RG.2.2.32593.45924.

[13] *Alber E-FIX propulsore di spinta elettrico per carrozzine manuali - Invacare Italy*. URL: https://www.invacare.it/it/carrozzine-elettroniche-scooters/propulsori-di-spinta-alber/alber-e-fix-propulsore-di-spinta-elettrico-carrozzine-manuali.

[14] *Alber E-PILOT moto propulsore per carrozzine manuali - Invacare Italy*. URL: https://www.invacare.it/it/carrozzine-elettroniche-scooters/propulsori-di-spinta-alber/alber-e-pilot-moto-propulsore-carrozzine-manuali.

[15] *Alber E-MOTION propulsore di spinta elettrico per carrozzine manuali - Invacare Italy*. URL: https://www.invacare.it/it/carrozzine-elettroniche-scooters/propulsori-di-spinta-alber/alber-e-motion-propulsore-di-spinta-elettrico-carrozzine-manuali.

[16] *Alber SMOOV ONE propulsore posteriore per carrozzine manuali - Invacare Italy*. URL: https://www.invacare.it/it/carrozzine-elettroniche-scooters/propulsori-di-spinta-alber/alber-smoov-one-propulsore-posteriore-carrozzine-manuali.

[17] A. C. Bertolino, A. De Martin, A. Nesci e M. Sorli. *Meccatronica - Analisi, progettazione e modellazione di servosistemi*. Set. 2021, pp. 303–306.

[18] *RADXA ROCK 4*. URL: https://rockpi.org/rockpi4.

[19] *SOLO MINI V2*. URL: https://www.solomotorcontrollers.com/specifications-solo-mini-v2-slm0322-4020/.

[20] Joseph P John, S. Suresh Kumar e B. Jaya. “Space Vector Modulation based Field Oriented Control scheme for Brushless DC motors”. In: *2011 International Conference on Emerging Trends in Electrical and Computer Technology*. 2011, pp. 346–351. DOI: 10.1109/ICETECT.2011.5760141.

[21] *SunFounder SF3218MG*. URL: https://www.sunfounder.com/products/20kg-high-torque-servo.

[22] *ESP32*. URL: https://www.espressif.com/en/products/socs/esp32.

[23] *Motion Terminal*. URL: https://www.solomotorcontrollers.com/SOLO-motion-terminal/2.7.1/.

[24] *ROS*. URL: https://www.ros.org.

[25] *ROS 2 Documentation: Foxy*. URL: https://docs.ros.org/en/foxy/index.html#.

[26] Valerio Cornagliotto, Michele Polito, Laura Gastaldi e Stefano Pastorelli. “Steering transparency control of a wheelchair assistive device based on state estimator approach”. In: The 16th IFToMM World Congress - November 5 Nishi-Shinjuku, Tokyo, Japan. In press.

[27] Valerio Cornagliotto, Michele Polito, Laura Gastaldi e Stefano Pastorelli. “Comprehensive Control Strategy Design for a Wheelchair Power-Assist Device”. In: 2023, pp. 162–170. DOI: 10.1007/978-3-031-32439-0_19.

[28] *Mraa library*. URL: https://github.com/eclipse/mraa.

[29] *SoloPy library*. URL: https://github.com/Solo-FL/SoloPy.