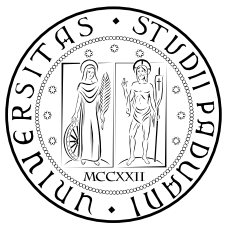

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCATRONICA

TESI DI LAUREA MAGISTRALE

QR01:
Progettazione e Controllo di un Sistema Attuato di Atterraggio per un Quadrirotore

Relatore: Giulia Michieletto

Correlatore: Stefano Michieletto

Laureando: Massimiliano Mocellin
1237240-IMC

ANNO ACCADEMICO: 2021-22

SOMMARIO

Questo lavoro di tesi è il pioniere del progetto QR01 il quale si pone l'obbiettivo di aumentare l'interazione di un drone quadrirotore con il mondo esterno. In particolare, questo elaborato, affronta un primo step di aumento dei gradi di libertà del velivolo e, nel dettaglio, la possibilità di atterraggio su superfici inclinate. Si è partiti con l'assemblaggio del velivolo vero proprio e con l'aggiunta dei dispositivi e dei software atti al volo indoor. Successivamente si è svolta un'analisi analitica di fattibilità per l'atterraggio su piani inclinati mediante l'aggiunta di un sistema di atterraggio attuato e di scelta della configurazione ottima da implementare, ricostruendo il modello del drone reale tramite CAD e progettando i nuovi pezzi. Sono stati sviluppati una serie di software in grado di controllare il nuovo sistema di atterraggio e di interfacciarsi, in futuro, con il drone reale. Per validare la soluzione progettata, si è creato un ambiente simulativo in cui è riprodotta la dinamica dell'intero sistema drone+sistema di atterraggio con particolare attenzione all'interazione tra i vari elementi, e al comportamento di attuatori e sensori. I risultati delle simulazioni svolte sono incoraggianti: il sistema di atterraggio progettato permette al quadrirotore di atterrare su piani con inclinazioni differenti evitando collisioni e ribaltamenti.

Il presente lavoro di tesi è stato svolto utilizzando esclusivamente software completamente open-source.

Spetta all'uomo ridare a sé stesso il dono della pace.

Robert Baden-Powell

RINGRAZIAMENTI

Gli anni della laurea magistrale sono stati intensi seppur brevi, intrisi di cambiamenti e conseguenti nuove abitudini. In questo periodo ho imparato quanto sia fondamentale la condivisione del tempo e degli spazi con gli altri e l'importanza dei legami. Per questo ci tengo a ringraziare tutte le persone che mi hanno permesso di affrontare più serenamente questa periodo e la monotonia delle giornate che sembravano non passare più. Ringrazio quindi tutti gli amici più cari, quelli che mi sono stati accanto durante questi tre anni e che mi hanno spronato a continuare. Ringrazio anche i nuovi amici, quelli che si sono affacciati nella mia vita proprio in questo periodo, quando sembrava che di socialità non si potesse parlare.

Ringrazio il mondo scout, per avermi accolto e per le esperienze che ho potuto fare. Sono state un importante fonte di crescita personale.

Ringrazio i relatori, per il supporto fornitomi e la possibilità concessami. Grazie a loro il mio percorso universitario ne é sicuramente uscito arricchito.

Infine, ovviamente, ringrazio infinitamente Giada per tutto quello che ha fatto per me e per la dedizione con cui mi ha aiutato a superare ogni ostacolo.

INDICE

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE

1 INTRODUZIONE

L'impiego di droni sta prendendo sempre più piede in un numero via via maggiore di settori, conseguenza del costo sempre più abbordabile dei componenti e del continuo sviluppo messo in gioco. Essi riescono a spostarsi in maniera rapida ed economica percorrendo distanze anche significative e a varie quote. Questo risulta interessante per quanto riguarda riprese di paesaggi o zone impervie, passando per un uso nelle situazioni di emergenza qualora altri mezzi su strada non riescano ad accedere ai luoghi interessati. Tutto ciò è semplificato dall'implementazione, oramai matura, di sistemi di agevolazione al comando e di guida autonoma, oltre alla sempre maggiore affidabilità in casi di perdita di segnali o malfunzionamento di sensori, portandosi quindi molto bene anche per l'automazione in diversi ambiti rivelandosi quindi efficienti anche in termini di autonomia. Con l'aumento della taglia dei droni e del numero di eliche, la massa trasportabile si aumenta notevolmente, permettendo anche il trasporto di merci in maniera autonoma e, soprattutto, economica. Nel mondo militare, inoltre, vengono ampiamente utilizzati in quanto permettono di sorvegliare delle aree senza mettere in pericolo il personale. Si sottolinea quindi come questo tipo di tecnologia stia diventando sempre più innovativa e rappresenti un settore in pieno sviluppo, evidenziato anche dall'incremento delle vendite anno per anno [9]. In questa situazione nasce l'esigenza di studiare nuovi sistemi sempre più avanzati per ampliare le capacità dei droni di interagire con il mondo esterno, non solo dal punto di vista di acquisizione di dati come già accade, ma piuttosto dal punto di vista del contatto e manipolazione di altri oggetti. Ci si concentra quindi sul volo all'interno di spazi chiusi e limitati, in cui la precisione nei movimenti è fondamentale e conseguenza di una conoscenza dettagliata della posizione del drone rispetto al mondo esterno. Si può proiettare successivamente il concetto di interazione con l'ambiente circostante verso un maggior controllo delle forze scambiate con altri oggetti e del posizionamento dello stesso in ambienti ristretti, permettendo al drone di compiere azioni tipiche di un robot.

1.1 IL PROGETTO QR01

Il progetto QR01 nasce dall'esigenza di migliorare genericamente l'approccio dei droni verso superfici non orizzontali, inteso sia come interazione, ovvero assegnando un task o un'azione specifica, sia come semplice atterraggio. Il progetto, in particolare, si focalizzerà inizial-

mente su quest'ultimo aspetto proseguendo man mano nel suo sviluppo e vedrà come protagonista un quadrotor di piccole dimensioni (HolyBro QAV250). A differenza di altri progetti presenti all'interno dello stesso laboratorio, QR01 si sofferma sull'addizione di gradi di libertà su questo drone a quattro eliche al fine di riuscire ad approcciare stabilmente superfici inclinate, cosa non possibile tipicamente con velivoli di questo tipo, creando delle strutture movimentabili che si adattano all'oggetto target. Il gruppo di ricerca nel quale rientra il presente progetto ha già sviluppato, in passato e in contemporanea a questa tesi, altri progetti legati ai droni, scegliendo di adoperare delle linee comuni in tutti i progetti, con lo scopo di rendere più efficiente il lavoro su nuovi aspetti. Seguendo questa filosofia è stato implementato un sistema di volo *indoor* necessario per permetterci di effettuare le prove all'interno della struttura a nostra disposizione in condizioni di assenza di segnale GPS e con l'obbiettivo di un'ottimizzazione della navigazione all'interno di spazi chiusi.

1.2 GLI OBBIETTIVI DELLA TESI

La presente tesi rappresenta il punto di partenza del progetto QR01 e comincia partendo dal drone QAV250 nella sua struttura di fabbrica. L'obbiettivo iniziale è di studiare e realizzare un generico sistema che permetta l'atterraggio su superfici inclinate tramite l'attuazione di un apposito meccanismo, da valutare analiticamente nella sua configurazione e struttura. La tesi, nello specifico, si concentra sullo studio di fattibilità e sulla progettazione di una prima versione del sistema richiesto. La progettazione, in particolare, riguarda sia la parte fisico-meccanica, sia la parte di controllo software, passando per implementazioni hardware di vario tipo. Il tutto viene implementato all'interno di un simulatore in modo da testare approfonditamente il funzionamento di ogni nuova implementazione.

Il risultato che si pone di ottenere il presente lavoro è di riuscire a realizzare un modello realistico del drone in ambiente simulativo in grado di decollare ed atterrare da e su piani inclinati ruotati in maniera arbitraria senza avere informazioni a priori riguardo l'ambiente esterno. Il sistema deve quindi essere autonomo nel ricavare le informazioni necessarie per svolgere la manovra in maniera efficiente e sicura.

1.3 STEP DEL LAVORO

Il lavoro svolto si suddivide in diverse fasi schematizzabili come segue:

- Studio dello stato dell'arte all'interno della pubblicazioni per capire quali sono gli studi già avviati in merito ed eventuali soluzioni già presenti. Ciò ha permesso di sviluppare una prima

idea sulle possibili strutture meccaniche e del sistema di sensoristica da utilizzare. Questa parte di lavoro non viene presentata all'interno della tesi in quanto ha solamente un ruolo di brain storming e quindi non è collegabile in maniera diretta a ciò che è stato progettato.

- Studio dei software quali PX4-Autopilot (impiegato all'interno del controllore di volo), Gazebo (come simulatore), e ROS2 (per la gestione della comunicazione). Tutto questo ha permesso una visione più specifica della soluzione da intraprendere e la scelta del software più opportuno per raggiungere gli obbiettivi prefissati.
- Montaggio del kit del drone, il quale viene venduto da assemblare, e prima configurazione per il volo.
- Creazione da del modello 3D tramite software CAD del QAV250 di fabbrica.
- Studio analitico delle configurazioni adottabili e conseguente ottimizzazione al fine di trovare una soluzione implementabile.
- Studio analitico della cinematica, utile nello sviluppo del software e del modello nel simulatore.
- Costruzione di una prima bozza del sistema finale nel progetto CAD.
- Calcolo e misura delle inerzie di ciascun pezzo. Conseguentemente si sono riportati tutti i valori all'interno del simulatore, aggiornando il modello anche con la nuova struttura aggiunta.
- Creazione del software per il controllo e la gestione che si interfaccino alla simulazione.
- Creazione del software per controllare il volo del drone ed interfacciarsi con il firmware a bordo.
- Verifica del funzionamento tramite test e valutazione dei risultati.

Parallelamente allo sviluppo appena presentato, si è svolto un lavoro di preparazione del drone fisico tramite la stampa 3D dei pezzi preparati ed il montaggio a bordo. Questo ha permesso di eseguire delle verifiche di compatibilità delle parti disegnate in CAD.

1.4 STRUTTURA DELLA TESI

Il seguito di questo elaborato è strutturato come segue:

- Nel capitolo 2 viene presentato inzialmente quello che è il drone utilizzato e i suoi componenti così com'è fornito di fabbrica, andando poi a presentare quelle che sono le componenti implementate in una prima fase di preparazione, sia dal punto di vista software che hardware.
- Il capitolo 3 discute analiticamente le prime fasi di studio di fattibilità che hanno portato ad una certa configurazione e i calcoli di cinematica effettuati.
- Il capitolo 4 affronta l'analisi simulativa di quello che è stato progettato, presentando la realizzazione del modello. Viene inoltre illustrato tutto il software che si interfaccia con la simulazione e che permette il funzionamento dell'intero sistema.
- Nel capitolo 5, invece, è raggruppato tutto il lavoro svolto per la realizzazione sperimentale e pratica di quanto presentato nei capitoli precedenti.
- Infine nel capitolo conclusivo si riporta sinteticamente quanto fatto commentando i risultato ottenuti.

2

SCENARIO APPLICATIVO: HARDWARE E SOFTWARE

Nel presente capitolo si introduce il modello di drone scelto per lo svolgimento della tesi. In particolare si analizzano i dispositivi di fabbrica compresi nel kit di assemblaggio e come sono stati assemblati. Ci si sofferma poi sul software di autopilota scelto e implementato all'interno del controllore del drone. Si presentano infine delle prime prove di volo effettuate per la taratura dei parametri, prima di procedere alla modifica della struttura.

Per perseguire gli obbiettivi di questa tesi, è stato necessario apportare delle modifiche hardware al QAV250 in modo da adattarlo alle nostre esigenze. A tal fine è stato posizionato a terra una mappa composta da dei particolari *tag* mentre a bordo del QAV250 sono stati montati una camera e un sistema di elaborazione. Quest'ultimo sarà in grado di elaborare le immagini video per andare a stimare la posizione di ciascun tag rispetto al sistema di riferimento del drone. Con opportune trasformazioni, si è quindi in grado di trasmettere al controllore di volo la posizione relativa del QAV250 rispetto alla mappa.

2.1 IL KIT HOLYBRO QAV250 + PIXHAWK4-MINI

Per lo svolgimento della tesi si è scelto di utilizzare il drone QAV250 della HolyBro. Si tratta di un medio-piccolo quadrirotore ad alte prestazioni, pensato per essere agile e dare buoni risultati nel volo acrobatico. Ciò è possibile grazie alle ridotte dimensioni e dallo scheletro costruito in fibra di carbonio che lo rendono particolarmente leggero. E' possibile acquistarlo scegliendo tra due kit, entrambi da assemblare, i quali comprendono tutti i pezzi necessari alla costruzione e utilizzo del drone completo, tranne la batteria e il radiocomando. I due kit differiscono per la presenza o meno della videocamera (FPV), utilizzabile per effettuare riprese e per la navigazione, installabile nella parte anteriore del drone.

Il kit acquistato (privo di videocamera) comprende:

- struttura in fibra di carbonio e relativa bulloneria;
- modulo gestione potenza HolyBro PM02, compreso di ESC già montati;
- dispositivo per la telemetria radio HolyBro Micro FPV;
- controllore di volo PixHawk 4 Mini e il relativo modulo GPS;

Figura 1: Foto dei pezzi della struttura presenti nel kit

- 4 motori brushless DR2205 KV2300;
- 4 eliche tripala da 5".

Oltre al kit sono stati acquistati anche:

- radiocomando RadioLink T9S Pro e ricevitore R9DS;
- batteria Lipo 3S.

2.1.1 *Struttura in fibra di carbonio*

La struttura del drone si presenta all'interno del kit come una serie di piastre in fibra di carbonio già opportunamente forate per il fissaggio con l'apposita bulloneria (Figura 1). Sono presenti ulteriori fori creati per minimizzare il materiale (e quindi il peso del drone) e per il passaggio di fascette, utilizzate quest'ultime per il fissaggio della batteria e altri accessori. In particolare le piastre sono così suddivise:

- Due piastre (in alto a sinistra in Figura 1), di dimensioni maggiori, che compongono la struttura centrale del drone, la quale ospiterà al suo interno il modulo di distribuzione della potenza

e il controllore di volo (Pixhawk). Questa struttura a sandwich è permessa per mezzo di appositi distanziali in alluminio.

- Quattro pezzi della medesima forma (in basso a destra in Figura 1), che collegano i motori alla struttura principale e costituiscono i bracci del drone.
- Un'ulteriore piastra, posizionata inferiormente (in alto a destra in Figura 1), per migliorare la resistenza alla flessione dell'intero assieme.
- Un piano, in materiale plastico, sul quale va incollata la Pixhawk (a destra al centro in Figura 1).

2.1.2 *Modulo gestione potenza HolyBro PM06 + ESC*

Il modulo di gestione della potenza è necessario nei casi in cui l'alimentazione provenga da una batteria e si hanno elevate correnti in gioco, in modo da salvaguardare la dinamica di scarica della batteria e aumentare la sicurezza dell'intero sistema. In particolare le funzioni principali svolte all'interno del drone sono:

- ricevere l'alimentazione della batteria tramite l'apposita presa e i cavi di elevata sezione;
- distribuire l'alimentazione ai 4 ESC, controllandone il consumo di corrente;
- convertire la tensione in ingresso, variabile in base allo stato di carica della batteria e alla corrente istantanea, in tensione 5V stabile per l'alimentazione delle periferiche;
- comunicare alla Pixhawk la corrente assorbita e la tensione della batteria tramite appositi segnali analogici.

Il modulo gestione potenza utilizzato (Figura 2), presenta una presa built-in per l'alimentazione della Pixhawk a 5V tramite l'apposito connettore. Due ulteriori contatti sulla PCB forniscono l'alimentazione a 5V, utilizzabili per collegare altri dispositivi quali servomotori o, come nel nostro caso, Raspberry Pi (che verrà presentato nei prossimi capitoli e viene impiegato per i calcoli ad alto livello) (Figura 3).

Sempre sulla superficie della PCB sono presenti altri contatti, due per ogni motore, alimentati direttamente alla tensione della batteria, ai quali sono collegati i 4 ESC (*Electronic Speed Controller*) [6].

L'ESC è un circuito in grado di pilotare motori di tipo *BLDCM* (BrushLess DC Motor), ovvero a corrente continua privi di spazzole, genericamente trifase e a bassa tensione. Nelle applicazioni come queste, in cui è richiesta un'elevatissima velocità di rotazione, si impiegano motori di tipo sensorless, ovvero privi di sensori in grado di

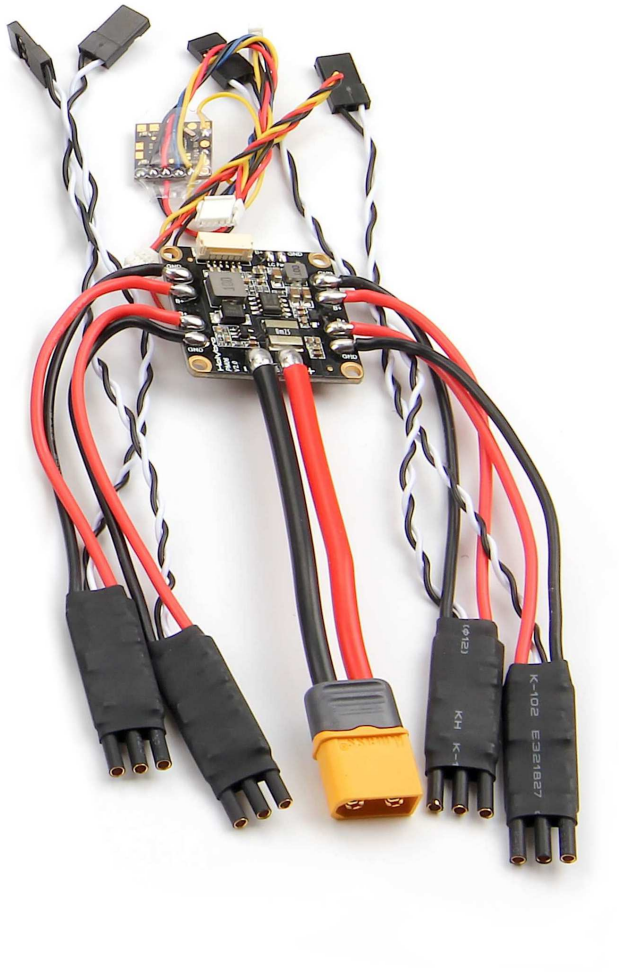

Figura 2: Foto del modulo di gestione potenza HolyBro PM06 con 4 ESC BLHeli S 20A ESC premontati

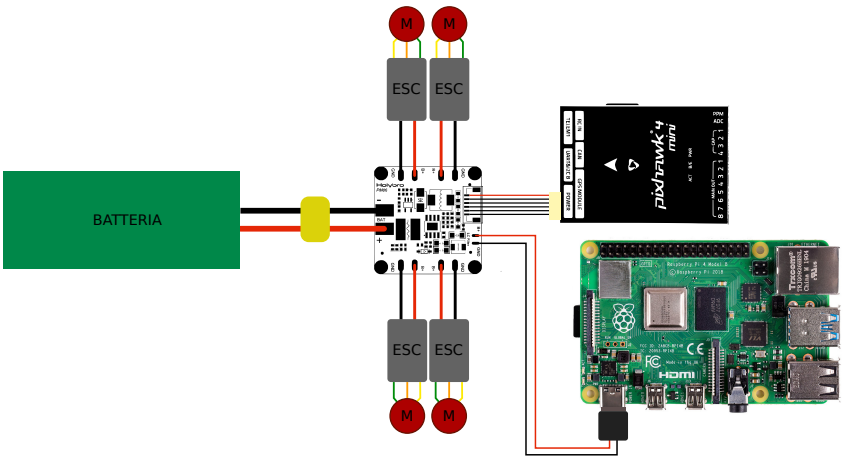

Figura 3: Schema dei collegamenti del circuito di alimentazione dei dispositivi

misurare la posizione del rotore. Perciò, per la connessione tra l'ESC e il motore sono sufficienti tre cavi di potenza, uno per fase, mentre la posizione del rotore, necessaria per pilotare correttamente le tre fasi, viene stimata o creata in maniera open-loop. Il circuito dell'ESC è sostanzialmente un piccolo inverter a commutazione digitale, pilotato da un apposito integrato in grado di effettuare le stime necessarie per l'azionamento e in grado di interfacciarsi con il controllore per mezzo di un apposito spinotto a 3 pin. Tra questi, la linea di segnale permette lo scambio delle informazioni tramite svariati protocolli, il più comune tra questi è il semplice PWM, ovvero un segnale digitale il cui periodo di commutazione è legato al target di velocità che si vuole trasmettere all'ESC. In alternativa, il OneShot si presenta come un segnale puramente analogico. Nei sistemi più avanzati è possibile comunicare tramite protocolli più complessi come il DShot, il quale trasmette i dati in maniera digitale e ad alta frequenza, ottenendo un miglioramento delle prestazioni e della risposta dell'azionamento [7]. In particolare, i vantaggi che questa tecnica porta in comparazione al PWM o al OneShot sono:

- non necessità di calibrazione e determinazione del valore di massimo e di minimo all'avvio;
- segnale meno sensibile ai disturbi (ad esempio quelli elettromagnetici provenienti dai motori posti nelle vicinanze) rispetto ai segnali analogici come il OneShot;
- risoluzione a 11 bit;
- trasmissione ad alta frequenza;
- facilità di rilevamento di eventuali dati corrotti, con conseguente aumento della sicurezza.

Dal momento che l'ESC in questione supporta tutte le tipologie di trasmissione appena citate, si è scelto di utilizzare il DShot300, dove la cifra sta ad indicare la velocità di trasmissione supportata (in questo caso 300000 bit/s). Tra le varie funzionalità, questo specifico modello di ESC permette di essere configurato in maniera piuttosto avanzata tramite l'interfacciamento con l'applicazione gratuita per Windows *BLHeliSuite32* [11]. Nel nostro caso, è stato utile l'utilizzo di questo programma per escludere il tono di *Beacon*, ovvero il segnale acustico che viene eseguito dopo alcuni minuti che l'ESC è acceso ma non utilizzato, attivo di default. Per l'interfacciamento del computer all'ESC si è utilizzato un Arduino Nano, collegando la linea dati del primo con uno dei pin di ingresso digitali del secondo. Infine, il collegamento di ciascun ESC con il relativo motore è reso possibile dai tre connettori a banana, uno per ogni fase. L'ordine di collegamento è poco significativo in quanto lo scambio di due fasi qualsiasi porterebbe solo ad un'inversione del senso di rotazione del motore. Tuttavia,

Figura 4: Foto dei due dispositivi di trasmissione telemetria radio HolyBro Micro FPV

in un secondo momento, è possibile reimpostare il verso di rotazione tramite appositi comandi previsti per la comunicazione in DShot.

2.1.3 *Telemetria radio HolyBro Micro FPV*

Si tratta di una coppia di dispositivi identici, utilizzabili per la trasmissione radio dei dati di telemetria (Figura 4). In particolare può essere utile fissare uno di questi alla struttura del drone collegandone l'apposita porta alla relativa presa "TELEMETRY" presente nella Pixhawk, mentre l'altro dispositivo può essere connesso tramite un comune cavo dati MicroUSB - USB Type-A ad un computer. Tramite applicazioni quali *QGroundControl* è quindi possibile visualizzare in tempo reale i dati di volo, modificare parametri all'interno del firmware del controllore di volo, scaricare i log di volo e molto altro. Nel nostro caso specifico, tuttavia, questi dispositivi sono stati rimpiazzati dal Raspberry Pi che si occupa, tra le altre cose, di effettuare questo tipo di collegamento (di tipo Mavlink) tra il controllore di volo e il computer.

2.1.4 *Pixhawk 4 mini + modulo GPS*

La Pixhawk 4 mini (Figura 5) è un controllore di volo avanzato, chiamato spesso anche autopilota, derivante dalla più comune Pixhawk 4 ma reso più compatto per poter essere utilizzato all'interno di droni di piccole dimensioni. Dalla Pixhawk 4 eredita lo stesso processore principale, ovvero l'STM32F765 e la stessa memoria RAM, mentre è

Figura 5: Foto della Pixhawk 4 mini (a destra) e del relativo modulo GPS (a sinistra)

assente l'*IO processor*. Al suo interno sono presenti svariati sensori, tra i quali:

- Accelerometro: in grado di misurare l'accelerazione lineare lungo i tre assi inerziali principali, permettendo quindi di stimare l'orientazione del drone nello spazio.
- Giroscopio: misura la velocità angolare rispetto agli stessi assi.
- Magnetometro (bussola): permette di conoscere l'orientazione del drone rispetto al campo magnetico terrestre e, in fase di setup, è in grado di quantificare il disturbo magnetico prodotto dai motori e dai relativi cablaggi di potenza.
- Barometro: misura la pressione atmosferica esterna, la quale viene utilizzata per stimare l'altitudine di volo.

Tramite le svariate porte presenti sul dispositivo è possibile collegare ulteriori periferiche e comunicare con il controllore di volo. Se ne riporta quindi una breve lista delle più utilizzate:

- Porta GPS MODULE: utile per collegare il relativo modulo GPS della Pixhawk, con il quale è possibile ricevere una stima della posizione assoluta del drone in termini di coordinate geografiche. Questo dispositivo è indispensabile nel caso si voglia far volare il drone in maniera autonoma, assegnandogli un tragitto da percorrere. Tuttavia questo non verrà impiegato perchè risulta troppo impreciso nei voli indoor e con aree di volo di dimensioni molto limitate.

- Porta TELEM1: con la quale è possibile scambiare dati di telemetria e, nel nostro caso, può essere collegato il modulo di telemetria radio HolyBro Micro FPV.
- Porta RC IN: necessaria per connettere un ricevitore per il radiocomando o simili.
- Porta POWER: solitamente collegata direttamente al modulo gestione potenza, il quale fornisce l'alimentazione a 5V stabili e comunica eventuali dati di gestione della potenza.
- Porte comunicazione seriale: sono presenti vari connettori attraverso i quali è possibile comunicare utilizzando svariati protocolli quali CAN, I2C, UART, SPI o aggiungere ulteriori periferiche (ad esempio sensori di stima della velocità o distanza dal suolo).
- 8 uscite principali: composte da 3 pin (V+ V- e segnale) che possono essere collegate, ad esempio, agli ESC o a eventuali servomotori.

Per il funzionamento del drone è possibile installare all'interno del controllore di volo un firmware che si occupa di gestire l'intera architettura, tra i quali si può ricordare i due più famosi *Ardupilot* e *PX4-Autopilot*, entrambi open-source. Nel nostro caso è stato scelto il secondo tra quelli appena citati PX4-Autopilot, il quale viene distribuito direttamente dal sito del produttore del Pixhawk. Esso è vantaggioso in quanto fornisce una documentazione molto dettagliata facilitandone l'impiego nella ricerca, settore in cui viene già ampiamente utilizzato.

2.1.5 *Motori brushless HolyBro 2206/KV2300*

Come già accennato, nell'ambito dei droni si utilizzano dei motori elettrici ad elevata densità di potenza e che riescono a raggiungere elevatissime velocità di rotazione. A tali scopi i BLDCM sono quelli che più si addicono, avendo le seguenti caratteristiche principali:

- Assenza di spazzole: porta ad una richiesta di minore manutenzione (essendo che le spazzole si usurano in breve tempo), minor rumore e minore attrito (non si ha lo strisciamento della spazzola).
- Presenza dei magneti permanenti rotorici: permette di ottenere elevate coppie con motori di dimensioni contenute e quindi poco pesanti.
- Ottima dinamica: in particolar modo come risposta a cambi repentini di target di velocità, dovuto alla bassa inerzia rotorica e alle elevate prestazioni.

Figura 6: Foto dei 4 motori brushless HolyBro 2206/KV2300 con relativa bulloneria per il fissaggio

- Elevata efficienza: è intrinsecamente più efficiente rispetto ad altri motori portando, di conseguenza, ad un minore energia termica prodotta che si traduce nella possibilità di incrementare le performance a parità di struttura o ad alleggerire gli accorgimenti atti alla sua dissipazione.

Inoltre, rispetto ai classici azionamenti BLDCM, spesso si rinuncia alla parte di sensoristica di posizione del rotore, utile al controllo dell'inverter per alimentare opportunamente le fasi, semplificando e alleggerendo così la meccanica del motore. Questa tipologia di motori, detta di tipo *sensorless*, può comunque essere pilotata da opportuni ESC in grado di stimare la posizione del rotore o, perlomeno, di offrire un controllo di tipo open-loop [1]. Come già specificato, e come nella quasi totalità degli azionamenti per droni, i motori sono di tipo trifase con centro-stella non accessibile e funzionanti a basse tensioni. E' infine utile precisare che tipicamente questi motori di piccola taglia presentano stampigliata nella scocca e/o nel codice modello del prodotto, una dicitura iniziante con "KV" e seguita da un numero. Tale dicitura rappresenta la costante di velocità equivalente del motore, riferita alla velocità espressa in *krpm* che esso può raggiungere a vuoto con un'alimentazione di 1 V picco-picco. Ciò permette di ricavare la velocità di rotazione massima ipotetica che si riesce a raggiungere a partire dalla tensione fornita dalla batteria. Questo dato è utile per la scelta delle dimensioni dell'elica [3]. In particolare i motori in uso presentano un valore di KV pari a 2300.

2.1.6 *Eliche*

Nel kit del QAV250 è compreso anche un set di eliche a tre pale di materiale plastico e diametro pari a 5"(Figura 7). Per ciascuna di es-

Figura 7: Foto delle 4 eliche impiegate

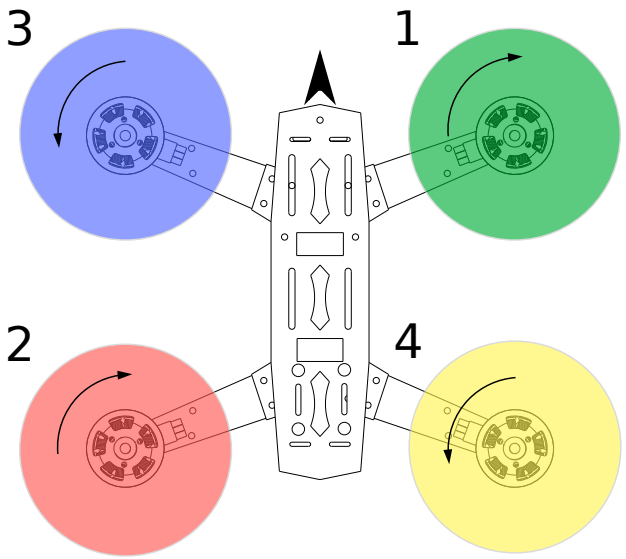

Figura 8: Schematizzazione del QAV250 con senso di rotazione delle eliche e numerazione dei motori

se è stampata, nella parte superiore, una freccia indicante il verso di rotazione previsto, che può essere orario (*CW*) o antiorario (*CCW*), e che dipende del verso di inclinazione delle pale. Questa informazione è utile in quanto il drone, per poter volare correttamente e per permettere il controllo del yaw[1], ha bisogno che le eliche ruotino in senso opposto rispetto a quelle adiacenti, pur mantenendo sempre la spinta prodotta orientata verso l'alto. Per queste ragioni, le eliche montate lungo una stessa diagonale dovranno avere lo stesso senso di rotazione, opposto rispetto a quelle montate nell'altra diagonale (Figura 8). L'impiego di eliche a tre pale anzichè due è dettato principalmente da due fenomeni aereodinamici:

- La spinta prodotta dalle eliche, a parità di dimensioni e velocità di rotazione, risulta quasi proporzionale al numero di pale (al netto dei fenomeni di interferenza). Si riesce quindi a produrre una spinta maggiore con delle eliche della stessa dimensione.
- Ottenendo una spinta maggiore si può ridurre il diametro dell'elica e di conseguenza la velocità massima periferica di ciascuna pala, riducendo quindi il rumore.

Ciascuna elica è fissata al rotore del motore tramite un dado autobloccante, che ne assicura la corretta aderenza al perno, mentre è consigliato smontarle, per questioni di sicurezza, durante l'esecuzione di prove di setup.

2.1.7 *Radiocomando RadioLink T9S Pro + ricevitore R9DS*

Venduto separatamente è possibile acquistare il radiocomando ed il relativo ricevitore da collegare all'apposita porta della Pixhawk (RC IN). Questo permette di pilotare il drone a distanza e di selezionare una delle varie modalità di volo messe a disposizione dal software. Nel nostro caso specifico il ricevitore modello R9DS della RadioLink permette sia il collegamento diretto degli ESC (comportandosi come controllore di volo meramente manuale) e di eventuale telemetria, sia l'impego del protocollo SBUS per la comunicazione digitale dei dati provenienti dal radiocomando e destinati ad un controllore di volo aggiuntivo, come nel nostro caso. L'accoppiamento con il radiocomando può essere effettuato tramite l'apposita funzione *bind*, attivabile

1 Nel linguaggio specifico aeronautico si usa nominare le tre rotazioni relative ai tre assi principali del velivolo come segue, riferite rispetto ad un osservatore orientato verso la parte frontale del veicolo stesso:

- roll (rollio), ovvero rotazione lungo un asse che va dalla punta alla coda;
- pitch (beccheggio), rotazione portando la parte frontale verso l'alto o verso il basso;
- yaw (imbardata), rotazione che sposta il muso a destra o a sinistra.

Figura 9: Foto del radiocomando RadioLink T9S Pro e specifica dei comandi più importanti

premendo il tasto dedicato presente nel ricevitore, la quale ricerca automaticamente il radiocomando posto nelle più immediate vicinanze e ne memorizza in maniera non volatile l'ID univoco. Per quanto riguarda il modello specifico di radiocomando scelto (Figura 9), essendo altamente configurabile, si omette la presentazione dettagliata delle caratteristiche e dei parametri impostati. Tuttavia, si riportano brevemente i comandi principali utilizzati e le funzionalità che sono state assegnate a ciascuno:

- *Throttle*: ha funzionalità variabile in base alla modalità di volo selezionata, ma genericamente serve a controllare l'altezza del drone.
- *Rudder*: permette di agire sullo yaw.
- *Elevator*: dipende dalla modalità di volo ma in generale agisce sul pitch e fa muovere in avanti il drone.
- *Aileron*: similmente al precedente, agisce sul roll.
- *Switch A*: permette l'armo/disarmo dei motori.
- *Switch B*: funziona come spegnimento di emergenza, ovvero spegne immediatamente tutti i motori a prescindere dallo stato del drone.
- *Switch C* e *Switch D*: selezionano la modalità di volo.

Ciascuno di questi comandi corrisponde ad un canale trasmesso al ricevitore, il quale può essere a sua volta assegnato ad una specifica funzionalità tramite il programma QGroundControl.

2.1.8 *Batteria*

La batteria utilizzata è della tipologia agli ioni di litio, scelta per l'elevata densità energia/peso che può raggiungere, rendendola tra le più adatte in questo ambito. E' composta da 3 celle, ciascuna con voltaggio nominale di 3.6 V, poste in serie. Il valore di tensione varia in base alla carica delle celle stesse, da un minimo di 9.6 V a batteria completamente scarica, fino ad un massimo di 12.6 V a piena carica. Questi valori devono essere rispettati rigorosamente in quanto un eventuale loro superamento può compromettere irreversibilmente la funzionalità della batteria, rendendola talvolta pericolosa. Per questo motivo il modulo di gestione potenza monitora continuamente la tensione della batteria per evitare che essa si scarichi oltre il limite imposto. La carica invece avviene per mezzo di un apposito alimentatore, il quale, oltre ad alimentare l'intero pacco batteria con la corrente adeguata, si occupa di bilanciare la tensione di ciascuna cella evitando che esse si sovraccarichino. Ciascuna batteria di questo tipo è caratterizzata da una sigla composta da un numero seguito dalla lettera "C". Questo valore indica il numero di volte che la batteria può essere scaricata completamente in un'ora ovvero rappresenta un'indicazione della corrente massima di scarica che la batteria può sopportare in relazione alla sua capacità complessiva. La corrente di carica, invece, spesso non è indicata ma si può ragionevolmente considerare pari ad un valore di 2 C. [8]

2.2 SOFTWARE PER IL CONTROLLO DEL VOLO

2.2.1 *PX4 Autopilot*

PX4 Autopilot è un software professionale *open-source* di controllo per veicoli senza pilota generalmente definiti "radiocomandati" quali droni, elicotteri, aerei, sottomarini, automobili, ecc. Si caratterizza quindi principalmente per la grande adattabilità a svariati impieghi. Per implementarlo è necessario compilare l'apposito firmware da installare nel controllore a bordo (ovvero la Pixhawk 4 mini in questo progetto), e impostare l'*airframe* tra una lista di predefiniti o, in alternativa, creandosi manualmente la configurazione del modello a partire da degli appositi file nominati *mixer*, i quali permettono di trasformare ciascun comando impartito dal controllore o dall'utente nel corrispondente comando agli attuatori. Lo stesso firmware si occupa principalmente della gestione di:

- attuatori (motori tramite ESC, servomotori, motori in corrente continua, luci, ecc);
- ingressi analogici (sensori);
- interfacce seriali di comunicazione con vari dispositivi;

- telemetria;
- ricezione radiocomando;
- informazioni di posizionamento, orientazione, altitudine;
- streaming video di camere;
- logging;
- controlli di sicurezza.

Nel caso dei droni, PX4 mette a disposizioni varie modalità di volo, sia assistite che autonome, selezionabili tramite il radiocamando o una stazione di controllo a terra, e che vanno a modificare il comportamento del controllore ai comandi del radiocomando. Le modalità implementate nel QAV250 sono:

- Per il controllo manuale tramite radiocomando:
  - *Position Mode*: quando gli *stick* del radiocomando sono in posizione centrale, il controllo cerca di mantenere il drone fermo in posizione. Un'azione sugli stick di elevator e aileron corrisponde ad un'accelerazione proporzionale rispetto al terreno, rispettivamente in avanti/indietro e sinistra/-destra. Lo stick del throttle, invece, controlla la velocità di ascesa/discesa.
  - *Altitude Mode*: l'azione degli stick è uguale alla modalità precedente ma, nel caso di stick in posizione centrale, viene controllata e mantenuta costante solo l'altitudine. Il drone sarà così libero di muoversi orizzontalmente.
  - *Stabilized Mode*: gli stick di elevator e aileron controllano l'inclinazione del drone, mentre il throttle la spinta verso l'alto dei propulsori. Il drone quindi risentirà di eventuali sbilanciamenti e della forza del vento, che dovranno essere compensati dall'utente.
  - *Acro Mode*: ciascun comando agisce direttamente sul mixer dell'output, e quindi va a variare la spinta per l'asse corrispondente al comando.
- Per il controllo automatico:
  - *Takeoff Mode*: porta il drone ad un'altezza impostata partendo da terra;
  - *Hold Mode*: mantiene la posizione e l'altitude impostata;
  - *Return Mode*: porta il drone verso una posizione sicura preimpostata (può essere attivata nel caso di perdita del segnale del radiocomando o altri problemi);
  - *Land Mode*: esegue un atterraggio dalla posizione attuale tramite un movimento verticale;

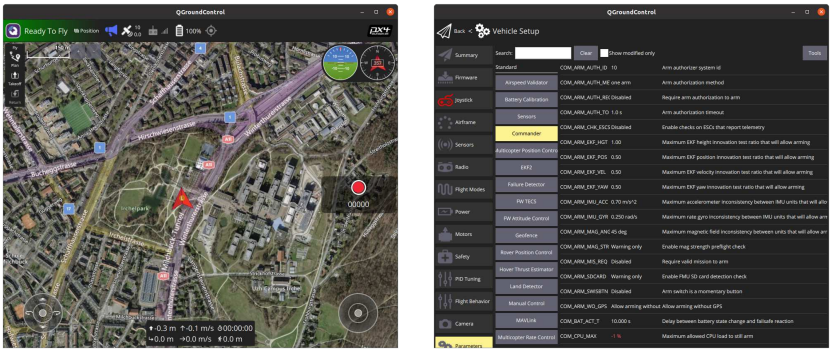

Figura 10: Schermate d'esempio del software QGroundControl. A sinistra la pagina iniziale, a destra il menù per la modifica dei parametri

- *Mission Mode*: segue un percorso preimpostato;
- *Follow Me Mode*: segue un certo track inviato;
- *Offboard Mode*: segue i setpoint inviati tramite MAVLink.

Si consideri tuttavia che avendo implementato un sistema di stima della posizione per il volo indoor, durante le prove di volo in laboratorio, al fine di testare correttamente l'intero sistema, vengono impiegate solo le modalità di volo Position, Altitude e Stabilized. Invece, per quanto riguarda la simulazioni, non avendo a disposizione il radiocomando e conseguentemente la possibilità di comandare il throttle, si impiegano le modalità di Takeoff e Land rispettivamente per far decollare e atterrare il drone.

PX4 è altamente configurabile in ogni sua parte utilizzando gli appositi parametri, di tipo numerico, modificabili tramite un programma di stazione di controllo a terra. Tra questi, il più famoso, è *QGroundControl*

2.2.2 *QGroundControl*

QGroundControl è un software per il controllo di volo open-source disponibile precompilato per Windows, Mac e sistemi Linux, ma anche per iOS e Android sotto forma di app. E' in grado di interfacciarsi con i controllori di volo che eseguono PX4 Autopilot oppure ArduPilot tramite svariate interfacce di comunicazione, quali ad esempio: connessione USB direttamente al controllore, dispositivi di telemetria radio e bridge MAVLink su protocolli UDP o TCP. Una volta connesso al veicolo, nella schermata principale (Figura 10 a sinistra) è possibile visualizzare i dati di odometria e la mappa del luogo in cui si trova. Da qui è possibile inoltre attivare una delle modalità di volo automatico ed impostare eventuali percorsi di Mission, come anche visualizzare lo stato della batteria e la modalità di volo attiva. Il menù, invece, dà la possibilità di configurare il controllore, ad esempio assegnando i tasti del radiocomando a ciascuna funzione, e di attivare

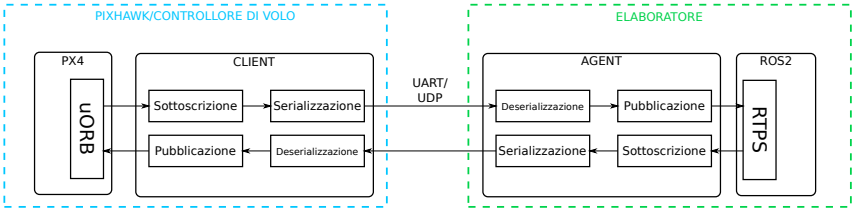

Figura 11: Schema comunicazione RTPS tra PX4 e un elaboratore

eventuali procedure di calibrazione dei sensori a bordo. Particolarmente importante è l'ultima scheda (Figura 10 a destra) che permette di variare i parametri e quindi di effettuare delle modifiche di tipo avanzato alla configurazione di volo e la possibilità di effettuare il download dei file di logging, contenenti i dati registrati durante il volo. In questo progetto QGroundControl è stato utilizzato principalmente per controllare il drone durante la simulazione (non potendo impiegare il radiocomando), esaminare i dati di volo, modificare parametri di PX4-Autopilot e ricavare i file di log delle sessioni di volo.

2.3 SISTEMA SOFTWARE AGGIUNTIVO

2.3.1 *ROS2*

ROS2 (Robot Operating System 2) è un insieme di strumenti software open-source e multi-piattaforma pensato per facilitare lo sviluppo nell'ambito robotico e industriale. Esso permette la creazione di nodi, ovvero algoritmi autonomi che si occupano di una specifica funzione, in grado di comunicare tra loro all'interno della stessa rete anche con nodi eseguiti su architetture hardware differenti. La comunicazione viene trasmessa tramite dei topic, ovvero dei collegamenti da un nodo *pubblisher* e uno o più nodi *subscriber*, utilizzando dei format di messaggi ben specifici. Per la gestione delle varie periferiche del drone e per l'implementazione degli algoritmi richiesti dallo studio, si è utilizzato ROS2 Foxy installato sul sistema operativo Ubuntu Server 20.04 LTS ed eseguito all'interno di un Raspberry Pi 4.

2.3.2 *microRTPS*

RTPS (*Real Time Publish Subscribe* protocol) è un'interfaccia di comunicazione utilizzata per interagire con PX4-Autopilot e pensata per essere leggera dal punto di vista computazionale. A differenza di altri sistemi di comunicazione a bordo del firmware, RTPS è particolarmente indicato nei casi in cui si necessiti di scambiare dati in maniera real-time e ad alta frequenza. In particolare il bridge microRTPS si occupa dello scambio di dati tramite una trasmissione seriale di tipo UART o UDP tra PX4 e un computer offboard. La tipologia di dati che si desidera inviare e ricevere possono essere specificati all'inter-

no di appositi file di configurazione con estensione .yaml, sia lato firmware che lato computer, i quali vengono impiegati al momento della compilazione. All'interno di PX4, microRTPS viene eseguito come processo client e si occupa di sottoscriversi e pubblicare messaggi di tipo uORB e di interfacciarsi con la comunicazione seriale tramite un apposita tecnica di pacchettizzazione di tipo generico (Figura 11). Il processo Agent, invece, eseguito lato computer, si interfaccia con i messaggi di tipo RTPS/DDS, sottoscrivendosi e pubblicando tramite appositi topic, anche di tipo ROS2.

2.3.3 *Visual Odometry Localization*

Con il termine Visual Odometry (VIO) Localization si intende quell'insieme di tecniche di posizionamento basate sulla fusione di misure inerziali (dati tipicamente provenienti da giroscopi e accelerometri) e informazioni di tipo visivo, in generale legate al rilevamento e all'identificazione di marker visivi. Questi ultimi possono essere di vario tipo e nel nostro caso si è scelto di utilizzare degli Apriltag ovvero un sistema di codice a barre di visione fiduciale a due dimensioni, molto simile al QRCode nella trama utilizzata. Viene impiegato in tutte quelle applicazioni in cui è necessario conoscere la posizione relativa di due oggetti utilizzando l'identificazione tramite camera. E' ottimizzato per poter funzionare anche con immagini a bassa risoluzione e l'algoritmo impiegato per il riconoscimento non richiede hardware particolarmente performanti. Gli Apriltag possono essere di varie famiglie, in base alla loro forma e dimensione, e possono essere generati tramite un apposito algoritmo open-source. Allo stesso modo, sono disponili svariati algoritmi per l'identificazione degli stessi [4]. I tag utilizzati nel sistema di localizzazione sviluppato in laboratorio sono della famiglia 41h12 e sono stampati sulla superficie di una mappa di dimensioni $5x7\ m$ posizionata a terra all'interno dell'arena di volo (Figura 12). Le dimensioni dei tag presenti è variabile e sono distribuiti in maniera omogenea all'interno dell'intera area stampata in modo da ottenere una stima accurata a diverse distanze del drone dal terreno. Per la creazione dell'intera mappa è stato sviluppato un apposito algoritmo, il quale è in grado anche di generare un file contenente le associazioni di ciascun tag, identificabile tramite un codice univoco, con la relativa posizione. Come già accennato, per il volo indoor è stato sviluppato un sistema in grado di rilevare degli appositi ArilTag tramite la camera a bordo del QAV250 e di ricavarne la posizione relativa rispetto al frame principale del drone. Per far questo un apposito nodo ROS, nominato *v4l2_camera_node*, si occupa di collegarsi alla camera, acquisendone le immagini, per poi pubblicarle in un apposito topic. Il nodo *apriltag_ros* quindi si sottoscrive a questo topic e, tramite un apposito algoritmo, è in grado di rilevare gli Apriltag presenti nelle immagini, ricavandone l'ID e la posizione relativa rispetto alla

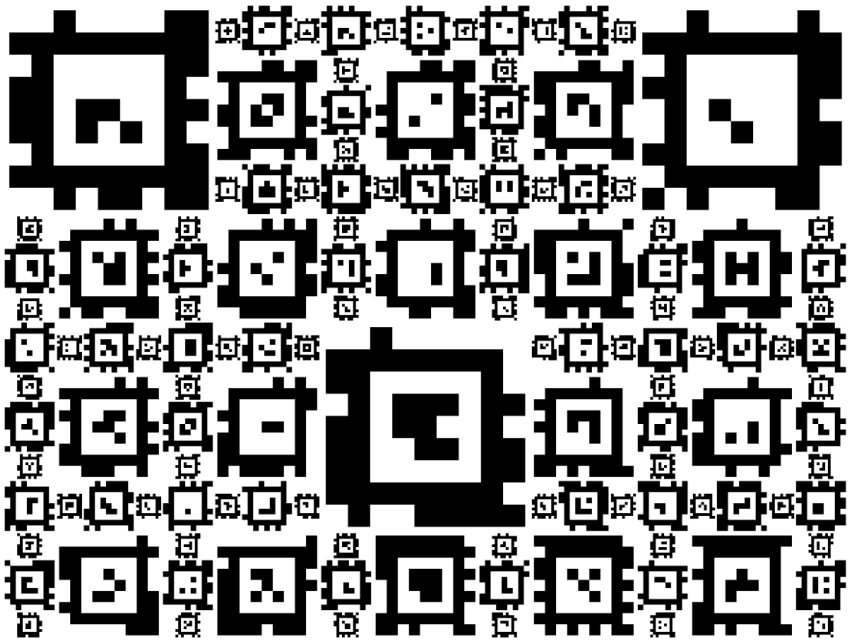

Figura 12: Esempio di mappa Apriltag utilizzata per il volo del QAV250

camera. Tutte queste informazioni vengono ulteriormente pubblicate e arrivano al nodo *apriltag_to_visual_odometry* il quale è in grado di ricevere in input uno specifico file di configurazione di tipo *yaml* che associa a ciascun codice di tag, presente nella mappa, la relativa posizione assoluta nello spazio. In questo modo, è possibile ricostruire la posizione assoluta del drone a partire dalla visione di uno o più apriltag presenti nella mappa. Tutte queste informazioni vengono inviate, tramite comunicazione RTPS, al firmware di PX4-Autopilot, il quale è in grado di gestire dati di posizione provenienti da varie sorgenti, quali barometro, magnetometro, GPS, ecc. filtrandoli e fondendoli tramite l'utilizzo di un filtro di Kalman esteso (EKF). Questo permette di ottenere un'unica informazione affidabile e precisa perchè basata su misure fornite da diversi sensori e stimata con modalità diverse.

Tuttavia, nonostante questo sistema sia stato implementato fisicamente a bordo del drone, essendo che gli obbiettivi del progetto si basano sull'uso di un simulatore, non verrà usato questo tipo di tecnica per stimare la posizione permettendo di alleggerire la complessità delle prove. Si utilizzerà invece le informazioni di posizione proveniente dal simulatore stesso e dagli eventuali plugin di simulazione dei sensori.

2.3.4 *Bridge MAVLink*

MAVLink è un protocollo di comunicazione creato principalmente per lo scambio di informazioni di telemetria nell'ambito dei veicoli senza pilota. Proprio per questo si adatta bene a implementazioni in

cui è richiesta una comunicazione di pochi dati ma da alte frequenze, con uno sfruttamento di risorse minimizzato. La comunicazione è principalmente di tipo multicast, caratteristica che permette un'alleggerimento dei pacchetti essendo questi privi di destinatario e generalmente non è garantita la consegna, utile per lo scambio dei dati di telemetria. Tuttavia, il protocollo MAVLink, prevede anche lo scambio di messaggi *point-to-point* e con consegna garantita, come per il caso di comandi o dati di missione. I dati all'interno del pacchetto vengono ordinati con degli algoritmi molto efficienti, dal più grande al più piccolo, in maniera da aumentare l'efficienza di trasmissione. Nello specifico del nostro progetto, MAVLink fornisce tutti i dati di telemetria provenienti dal controllore di volo e che si vogliono visualizzare tramite QGroundControl sulla stazione di controllo (tipicamente un computer). Inoltre, questo tipo di comunicazione permette una rapida modifica dei valori di PX4 senza dover accedere manualmente al firmware. Per consentire questo scambio di informazioni in volo e senza l'ausilio di collegamenti via cavo, è possibile utilizzare il dispositivo di telemetria radio fornito all'interno del kit. Tuttavia, essendo già presente un elaboratore connesso alla rete wireless e alla porta seriale della Pixhawk, si fa uso di un bridge MAVLink installato direttamente nel sistema operativo presente sul Raspberry Pi. Questo si occuperà di tradurre e trasmettere i pacchetti MAVLink tramite la rete LAN IEEE 802.3 e utilizzando il protocollo UDP.

2.4 HARDWARE AGGIUNTIVO

2.4.1 *Raspberry Pi 4*

L'elaborazione richiesta dai nodi descritti nella sezione 2.3.1, è affidata ad un Raspberry Pi 4 Model B, un piccolo computer a scheda singola che presenta le seguenti caratteristiche:

- processore: architettura RISC, Quad core Cortex-A72 (ARM v8) 64-bit;
- RAM: 8GB LPDDR4-3200 SDRAM;
- WLAN: 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE;
- LAN: Gigabit Ethernet;
- USB: 2 porte USB 3.0, 2 porte USB 2.0;
- memoria: tramite scheda microSD da 128 GB;
- alimentazione: 5V CC tramite porta USB-C o pin.

Per poterla montare a bordo del QAV250 in maniera salda, fissandola alla piastra superiore della struttura, si è disegnato un apposito

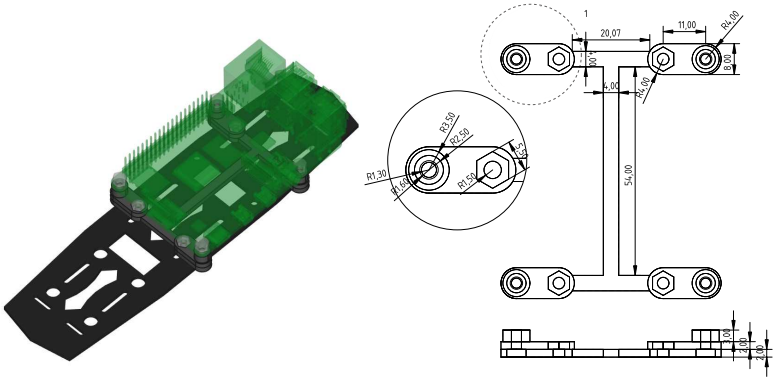

Figura 13: Rendering 3D del supporto per il Raspberry Pi applicato al QAV250 (a sinistra) e relativo disegno tecnico (a destra)

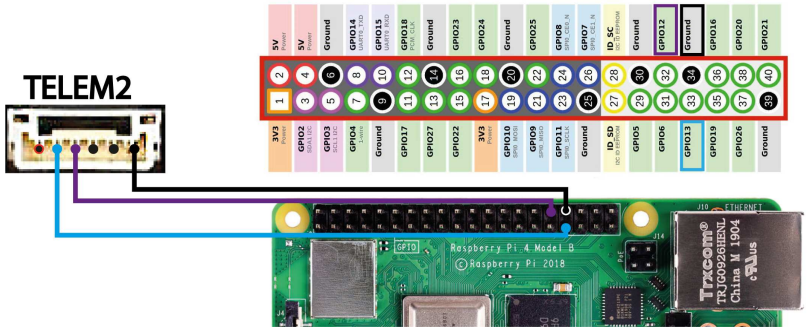

Figura 14: Schema della connessione tra Pixhawk 4 mini e Rapsberry Pi per lo scambio di messaggi microRTPS su connessione seriale

adattatore (Figura 13). Questo va ad utilizzare le scanalature già presenti sulla piastra in fibra di carbonio per adattarle ai fori sulla PCB del Raspberry, situati in posizioni diverse. La scelta di utilizzare proprio le scanalature longitudinali è dovuto alla possibilità di variare la posizione di fissaggio del pezzo, andando così a bilanciare i pesi dell'intero drone. Il pezzo è stato poi stampato, tramite l'ausilio di una stampante 3D, in materiale plastico (ABS). Dalla figura si può osservare che, per agevolare il montaggio, ciascun foro presenta una struttura con una forma esagonale inscritta, la quale alloggerà e bloccherà dalla rotazione il dado dei bulloni di fissaggio.

Per l'alimentazione del Raspberry si è sfruttato il modulo di gestione potenza, il quale presenta di fabbrica dei contatti aggiuntivi per l'alimentazione di periferiche a 5V, sui quali è stato saldato un cavetto tramite stagnatura. Sull'altra estremità del cavo è già premontata una porta USB-C la quale va collegata direttamente alla relativa porta di alimentazione del Raspberry. Per quanto riguarda lo scambio di informazioni tra ROS2 e PX4, si è impiegato un bridge microRTPS, realizzato tramite connessione di porte seriali UART. Dal punto di vista fisico, si è partiti da un cavo JST GH 6P modificandone una delle due estremità per poterle connettere ai pin del Raspberry Pi sulla

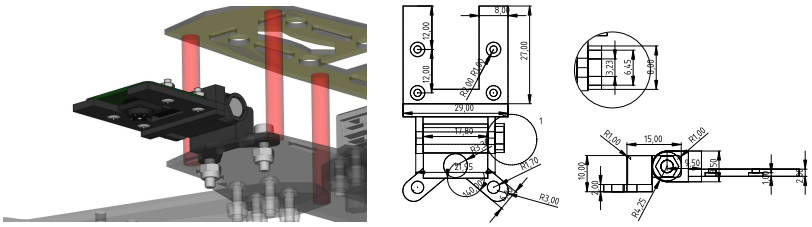

Figura 15: Rendering 3D del supporto della camera applicato al QAV250 (a sinistra) e relativo disegno tecnico (a destra)

porta UART5 (Figura 14). L'altra estremità invece è stata connessa al TELEM1 della Pixhawk. Un'ulteriore collegamento si è reso necessario per la comunicazione MAVLink con il controllore di volo, spiegata in sezione 2.3.4. Per far questo si è impiegato un cavo con una spina di tipo USB-A da un lato, collegata ad una porta USB del Raspberry, e dall'altro una porta USB Micro-B inserita nell'apposita presa seriale della Pixhawk.

2.4.2 *Camera*

La camera utilizzata per l'acquisizione delle immagini è una Pi Camera Module V2 NoIR, ovvero una fotocamera da 8 megapixel progettata appositamente per essere collegata al Raspberry Pi tramite l'apposita porta Camera presente sulla superficie della PCB, e quindi ottimizzata per l'uso su questo specifico elaboratore. La sigla "NoIR" sta ad indicare l'assenza di un filtro infrarossi, e quindi la possibilità di acquisire anche questo tipo di lunghezze d'onda. Come per il Raspberry, per il fissaggio della camera alla struttura del drone è stato progettato e stampato in 3D un supporto ad hoc, avendo le stesse accortezze dell'adattatore per il Raspberry (Figura 15). Il sistema operativo installato sul Raspberry Pi è Ubuntu Server 20.04 LTS ARM64, una distribuzione Linux molto leggera priva di interfaccia grafica, nella sua versione *Long Term Support* adattata per processori di tipo ARM a 64 bit. Per poter comunicare con la stazione di controllo a terra e con eventuali interfacce si è connesso il Raspberry Pi alla rete LAN tramite la connessione Wifi presente di serie, la quale permette uno scambio di grandi quantità di dati agevolmente ma limitatamente all'area dell'arena di volo e quindi non pensato per essere utilizzato anche per voli outdoor. La stessa connessione ci permette di accedere da remoto al terminale di Ubuntu Server tramite protocollo SSH, evitando di doversi collegare manualmente con uno schermo e una tastiera.

2.4.3 *Calibrazione della camera*

Ogni camera presenta delle distorsioni dell'immagine acquisita dovuta alle lenti e ad eventuali non idealità di costruzione della stessa, che quindi devono necessariamente essere corrette nelle situazioni in cui si voglia ricavare ogni sorta di misurazioni tramite l'acquisizione di immagini. Per questo motivo si ricorre ad un procedimento di calibrazione, il quale è in grado di ricavare eventuali parametri di distorsione caratteristici di una certa specifica camera tramite l'ausilio di geometrie note all'interno dello spazio 3D visibile dalla camera. [12] Nel nostro caso specifico, è stato utilizzato un nodo ROS2 denominato *camera_calibration* eseguito su un PC, il quale va a sottoscriversi al topic del nodo *v4l2_camera_node*. Esso presenta un'interfaccia grafica che permette di monitorare l'avanzamento della calibrazione e, al termine, di salvare i parametri all'interno di appositi file. Questi saranno poi inseriti in apposite cartelle e verranno utilizzati dagli altri nodi per la correzione delle distorsioni. Il procedimento di stima richiede l'impiego di un supporto rigido bianco sul quale è stampata in nero una delle tante geometrie utilizzabili, nel nostro caso una scacchiera 8x10. Questa deve essere posizionata di fronte alla camera in maniera da essere completamente visibile e va mossa in maniera casuale e lungo tutti i gradi di libertà, preoccupandosi che sia sempre visibile all'interno dell'inquadratura. Il nodo di calibrazione esegue tramite il comando:

```
$ ros2 run camera_calibration cameracalibrator --size
   =7x10 --square=0.05 --pattern='chessboard' image
   :=/camera/image camera:=/camera
```

nel quale gli argomenti stanno ad indicare:

- *size*: dimensione della scacchiera contando solo i quadrati interni (si escludono quindi quelli più esterni);
- *square*: dimensione in metri di ciascuna casella;
- *pattern*: tipologia di geometria utilizzata,
- *image*: topic del nodo camera;
- *camera*:

[13]

2.4.4 *TOF*

Generalmente, con TOF (Time Of Flight) si intende una famiglia di sensori in grado di rilevare la distanza di un oggetto target opaco in maniera particolarmente accurata. Il loro funzionamento si basa principalmente sull'emissione di fotoni sotto forma di raggi luminosi

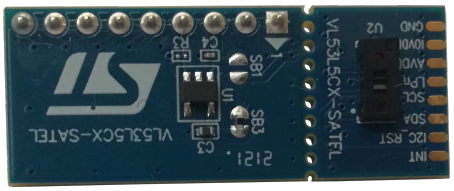

Figura 16: Foto del TOF STMicroelectronics NV VL53L5CX-SATEL in uso

ad impulsi, i quali rimbalzano sulla superficie target più prossima, per venire catturati dal sensore. Tramite la misura del tempo di volo del fotone, ovvero il tempo trascorso tra l'emissione e la ricezione dello stesso, si può stimare la distanza dell'oggetto che l'ha riflesso. I pregi di questo sistema rispetto ad altri sensori impiegabili per i medesimi scopi sono:

- bassa dipendenza dalle caratteristiche del target, quali colore, materiale ecc.;
- buona precisione anche ad elevate distanze;
- laser non visibile ad occhio nudo;
- bassissimo consumo.

Questa tipologia di sensori si può trovare spesso nel formato di integrati all-in-one, ovvero implementanti di fabbrica al loro interno l'intero sistema, senza richiedere particolari configurazioni, e che forniscono in uscita le letture in formato digitale. [19] Per quanto riguarda il progetto del QAV250, è necessario l'impiego di sensoristica di questo tipo per la rilevazione della distanza dal piano di atterraggio e per misurarne la pendenza. In particolare sfrutta un TOF della STMicroelectronics NV (azienda produttrice di componenti elettronici) in grado di effettuare la misurazione su più zone (per un totale di 8x8). Questo sensore, montato nella parte anteriore del drone rivolto verso il basso, viene venduto, oltre che nella sua forma all-in-one con codice *VL53L5CX*, anche già premontato su delle pcb nella sua versione *VL53L5CX-SATEL*. In quest'ultimo formato risulta più facile il montaggio a bordo e presenta già una piedinatura più comoda per la saldatura dei collegamenti. L'alimentazione richiesta è a tensione continua di 3.3 V, ricavabile direttamente dai pin del Raspberry e senza necessitare di convertitori, mentre la comunicazione digitale dei dati è affidata al protocollo seriale I_2C. Ciascuna zona può misurare distanze comprese tra i 2 cm e i 400 cm

3

SISTEMA DI ATTERRAGGIO: STUDIO E PROGETTAZIONE

Il progetto qui presentato, nominato all'interno del nostro laboratorio di ricerca come QR01, nasce dall'obbiettivo comune di progettare dei sistemi di volo autonomo in grado di interagire con l'ambiente esterno. Si pensi, ad esempio, all'afferraggio di pezzi, all'applicazione controllata di forze su superficie di varia inclinazione, all'atterraggio su piani inclinati, ecc. Ad ora lo stesso laboratorio ospita anche un altro progetto parallelo, nominato HR01, il quale impiega un esarotore di dimensioni molto maggiori del QAV250. Entrambi i progetti, nel breve periodo, si stanno occupando dell'applicazione del volo indoor (tramite Apriltag, già presentato nel capitolo precedente) e all'incremento dei gradi di libertà controllabili del veicolo stesso. Nel caso di un drone a 4 eliche, infatti, gli attuatori presenti e controllabili in maniera indipendente sono 4 e, di conseguenza, i gradi di libertà che si possono pilotare risulteranno essere al massimo 4, ovvero roll, pitch, yaw e altitudine. La traslazione destra-sinistra o avanti-indietro si ottiene tramite il controllo rispettivamente del roll e del pitch, ovvero inclinando il drone in una certa direzione in maniera proporzionale all'accelerazione che si vuole ottenere. Non è quindi possibile controllare indipendentemente alcuni di questi movimenti, non permettendo al drone, ad esempio, di rimanere inclinato e fermo. Semplificando, questo fenomeno è dovuto alla forza prodotta dalle eliche che è sempre solidale alla struttura del drone. Questa caratteristica può essere ovviata nei casi di droni con numero maggiore di attuatori e tramite particolari accortezze, ma, nel caso del quadrotor, la soluzione intrapresa si basa sull'utilizzo di un meccanismo aggiuntivo, il quale permette di approcciare superfici inclinate mantenendo il drone stabile e orizzontale. Si precisa però che, nel seguito della trattazione, i termini *superficie di atterraggio*, *piano inclinato*, *superficie inclinata* o simili, sono impiegati come sinonimi ed indicanti una generica superficie che il drone deve approcciare.

Nel presente progetto si sono affrontate inizialmente delle ricerche bibliografiche, allo scopo di trovare delle soluzioni potenzialmente applicabili e per conoscere approfonditamente il funzionamento del sistema utilizzato (come, ad esempio, il firmware PX4 ed il software ROS2). A partire da una prima bozza schematica degli obbiettivi, sono seguiti una serie di studi analitici per trovare la configurazione ottima da scegliere.

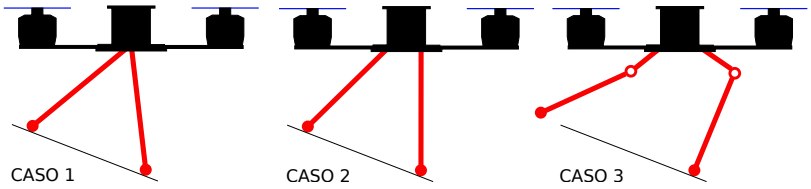

Figura 17: Rappresentazione stilizzata delle zampe nelle tre configurazioni studiate

3.1 ANALISI PRELIMINARE DELLE POSSIBILI SOLUZIONI

In questa prima fase del più ampio progetto QR01, ci si concentra sullo studio e sulla costruzione di un sistema di atterraggio attuato, in grado di adattarsi alla pendenza del terreno sottostante. In questo modo il drone dovrà essere in grado, entro determinati limiti, di atterrare sulla superficie di piani inclinati, ipotizzati inizialmente essere uniformi e di estensione illimitata. Per permettere di eseguire uno studio a livello monodimensionale, le superfici dovranno essere inclinate verso un'unica direzione predefinita.
Si sceglie di impiegare una coppia di pattini di atterraggio, che denominiamo "zampe", in grado di ruotare lungo un perno fissato alla struttura del drone, tramite un apposito motore controllato tramite Raspberry Pi. Lo studio quindi si articola in un prima scelta della configurazione ottimale tra le tre possibili (Figura 17):

- Caso 1: le zampe hanno gli assi di rotazione coincidenti, ovvero utilizzano uno stesso perno per ruotare.
- Caso 2: gli assi di rotazione sono distinti ma posizionati lungo la base del drone.
- Caso 3: gli assi di rotazione sono distinti e traslati più in basso rispetto alla base, ovvero il meccanismo presenta una sorta di ginocchia.

Per determinare quale di queste configurazioni è la più efficiente, è necessario valutare caso per caso qual è la pendenza massima ammissibile al variare dei vari parametri geometrici. Tuttavia, per il calcolo del limite massimo di inclinazione, i fattori che consideriamo sono tre e tali da dover essere rispettati tutti contemporaneamente:

- Ingombro: nessuna delle zampe deve collidere con la struttura del drone stesso.
- Ribaltamento: la proiezione del baricentro del drone sulla superficie di atterraggio deve stare all'interno dei punti di appoggio, evitando quindi delle condizioni di ribaltamento.
- Collisione: la struttura del drone non deve collidere con la superficie di atterraggio.

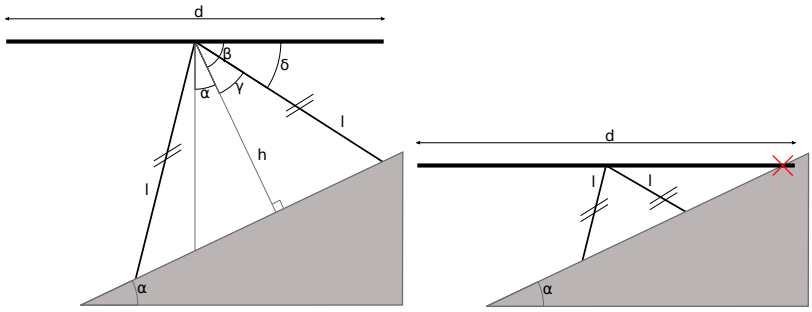

Figura 18: Disegno semplificato del drone nella configurazione del caso 1 con dimensioni geometriche impiegate nella trattazione (a sinistra) ed esempio di collisione nel caso di $l < d/2$ (a destra)

Il limite complessivo sarà il valore di pendenza più basso che rispetti le condizioni appena elencate in funzione dei parametri. Lo stesso studio, oltre a fornire un'idea delle prestazioni ottenibili per ciascuna delle configurazioni precedenti, ha lo scopo di ottimizzare la scelta dei parametri a partire da delle condizioni obbiettivo definite a priori. Per permettere una trattazione di tipo analitico è stato necessario considerare un modello geometrico semplificato, il quale considera l'intero drone come un unico piano con estensione peggiorativa rispetto all'ingombro effettivo del modello reale considerando anche le pale dei rotori. Inoltre, si assume che il baricentro del drone appartenga a questo piano e che sia posizionato in posizione centrale.
Partendo da queste ipotesi è possibile studiare caso per caso, andando a ricavare i limiti richiesti.

3.1.1 *Caso 1: Assi coincidenti*

A partire da una semplificazione bidimensionale del caso come in Figura 18 a sinistra, si considerano le seguenti dimensioni:

- β: è l'angolo compreso tra il piano principale del drone e la retta passante per il suo centro di massa perpendicolare al piano inclinato h;
- α: è l'angolo tra h e l'ortogonale al piano principale;
- γ: è l'angolo compreso tra la zampa destra e h;
- δ: è l'angolo compreso tra il piano principale e la zampa sinistra;
- l: la lunghezza delle zampe;
- d: la dimensione massima dell'ingombro.

Per costruzione è possibile effettuare le seguenti operazioni tra angoli (Figura 18 a sinistra):

$$\beta = 90° - \alpha \tag{1}$$

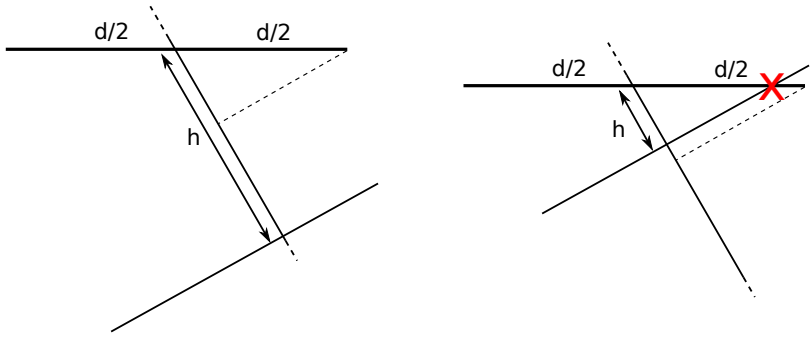

Figura 19: Esplicazione geometrica della disequazione di collisione nel caso 1

$$\delta = \beta - \gamma = 90^\circ - \alpha - \gamma \tag{2}$$

Il limite di ingombro impone che $\delta \geq 0$ e quindi:

$$\alpha \leq 90^\circ - \gamma \tag{3}$$

Il limite di ribaltamento richiede che $\alpha \leq \gamma$ per evitare che la proiezione del baricentro esca dalla base di appoggio creata dalle zampe. Per la collisione bisogna distinguere il caso in cui le zampe sono abbastanza lunghe da non permettere l'impatto tra il piano del drone e la superficie inclinata:

- caso $l \geq d/2$: l'impatto non avviene mai, le zampe sono troppo lunghe
- caso $l < d/2$ (e $l > 0$):

  $$h = l \ \cos\gamma \tag{4}$$

  Dalla figura 19 è evidente che le zampe devono essere lunghe a sufficienza per garantire che h sia maggiore o uguale alla proiezione di $d/2$ sull'asse su cui giace h, ovvero l'asse ortogonale alla superficie di atterraggio. La condizione di non collisione è quindi:

  $$h \geq \frac{d}{2}\cos(\beta) = \frac{d}{2}\sin(\alpha) \tag{5}$$

  Quindi, ricavando α:

  $$\alpha \leq \arcsin\left(\frac{2l}{d}\cos\gamma\right) \tag{6}$$

3.1.2 *Caso 2: assi distinti*

Per lo studio del limite di ingombro, ci si riconduce al caso critico in cui una delle zampe è posizionata in orizzontale, parallela al piano

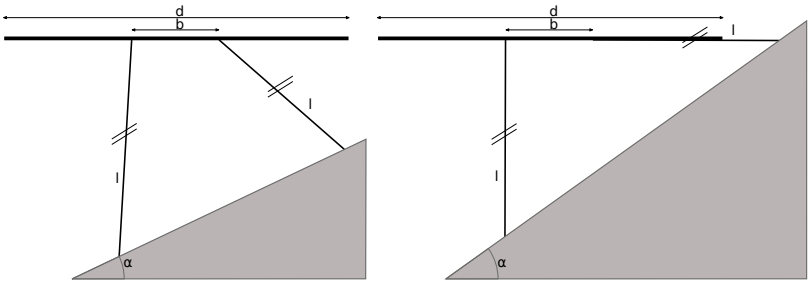

Figura 20: Disegno semplificato del drone nella configurazione del caso 2 con dimensioni geometriche impiegate nella trattazione, in situazione generica (a sinistra) e in condizione limite (a destra)

del drone e quindi collidente con lo stesso (Figura 20 a destra). L'altra zampa, per massimizzare la pendenza del piano inclinato e per evitare condizioni di instabilità, viene ipotizzata essere in posizione verticale.

Si definisce inoltre, come da Figura, le dimensioni:

- b: distanza tra i perni delle zampe;
- l: lunghezza di ciascuna zampa;
- d: la dimensione massima dell'ingombro.

A partire da queste ipotesi, si può utilizzare il teorema del seno come segue:

$$\frac{l}{\sin\alpha} = \frac{b+l}{\cos\alpha} \tag{7}$$

Moltiplicando entrambi i membri per $\frac{\sin\alpha}{b+l}$ si ottiene:

$$\tan\alpha = \frac{l}{b+l} \tag{8}$$

che rappresenta il caso critico. Riportando l'equazione sotto forma di intervallo di non collisione:

$$|\alpha| \leq \arctan\left(\frac{l}{b+l}\right) \tag{9}$$

Il ribaltamento, a partire dalle ipotesi precedenti, è sempre escluso. Anche la collisione con la superficie non avviene mai per $d < b + 2l$, condizione sempre verificata nella pratica e quindi esclusa dallo studio.

3.1.3 *Caso 3: assi distinti e presenza di ginocchia*

Questo caso può essere ricondotto al precedente, considerando la presenza del ginocchio come una semplice traslazione degli assi di rotazione, la quale può essere scomposta nelle due componenti in x e y.

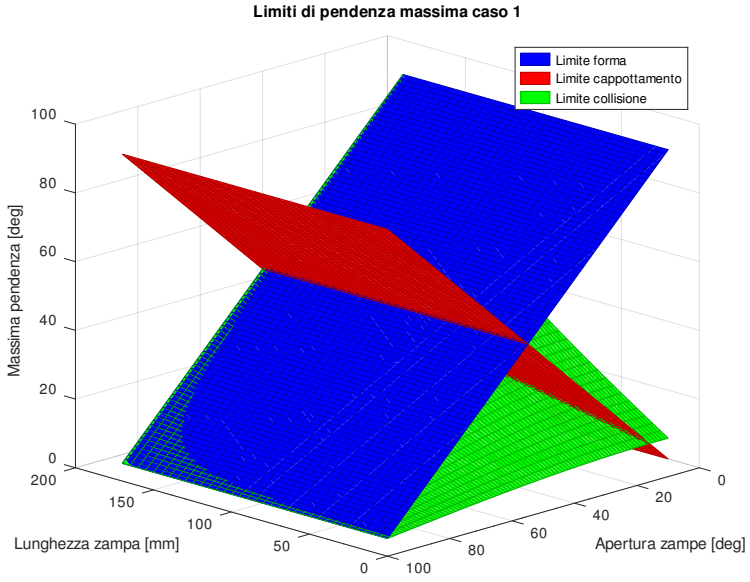

Figura 21: Limiti di pendenza massima raggiungibile in funzione della lunghezza delle zampe e dell'apertura delle zampe

La traslazione lungo l'asse x è equivalente ad un aumento del valore di *b* del caso 2, mentre la traslazione in y, invece, porta a dei benefici in termini di limite di collisione e di forma rispetto al caso precedente. Essendo, il presente studio, svolto al fine di scegliere quale delle configurazioni presentate in precedenza è più efficace, si può considerare inutile trattare in maniera analitica anche questo caso che, come è evidente, risulta essere una miglioria del caso 2.

3.1.4 *Conclusioni*

Per l'attuale progetto si considera come obbiettivo l'atterraggio su superfici con inclinazione massima di 25°, permettendo così di trovare dei parametri ragionevoli riguardo la lunghezza delle zampe e la loro disposizione. Utilizzando un algoritmo GNU Octave, è quindi possibile verificare che, confrontando i primi due casi, il secondo riesce a raggiungere l'obbiettivo di inclinazione massima impiegando delle zampe di lunghezza inferiore (Figure 21 e 22). Questa caratteristica è fondamentale nell'applicazione considerata in quanto zampe troppo lunghe possono peggiorare di molto la dinamica del drone in volo, creando un momento d'inerzia troppo elevato da gestire e aumentando di molto la massa dell'intero velivolo. Inoltre, zampe più lunghe, a parità di sezione, saranno più fragili e quindi meno resistenti ad impatti col terreno in fase di atterraggio. Non si considera tuttavia necessario ricorrere al terzo caso in quanto la base del drone risulta già sufficientemente larga da ospitare i perni e non si ritiene oppor-

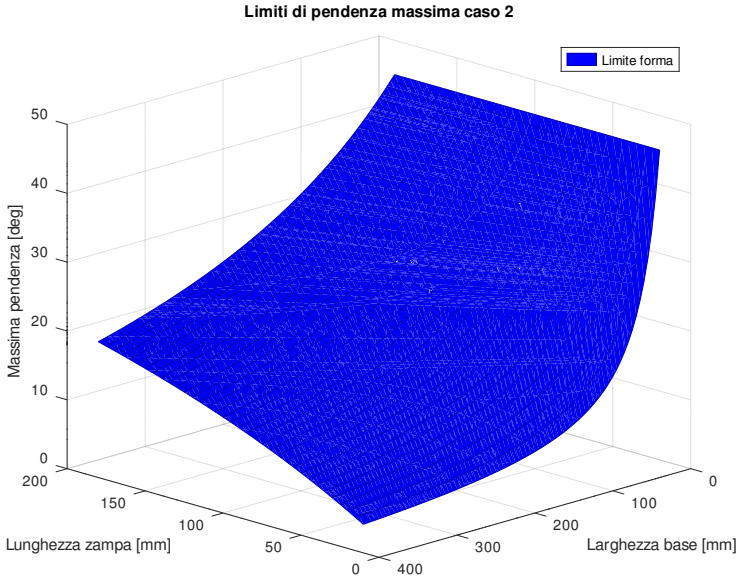

Figura 22: Pendenza massima raggiungibile in funzione della lunghezza delle zampe e della distanza tra i due assi di rotazione nel caso 2

tuno complicare la geometria del meccanismo in modo da ospitare ginocchi. A partire dai dati ricavati è possibile estrarre una configurazione che permetta di raggiungere l'obbiettivo di pendenza prefissato ovvero, seguendo la figura 22, un punto che stia al di sotto delle tre curve tridimensionali e che abbia una coordinata in z superiore o uguale a 25°. Questo punto dovrà inoltre minimizzare la lunghezza delle zampe. I valori quindi scelti sono:

$$l = 0.107\ m \qquad b = 0.050\ m$$

3.1.5 *Scelta numero di azionamenti*

Ipotizzando che ciascuna delle due zampe possa ruotare attorno al relativo perno controllate da Raspberry Pi tramite l'introduzione di appositi attuatori, l'obbiettivo del progetto è l'aggiunta di un grado di libertà, ovvero quello relativo all'inclinazione del piano di contatto col terreno. In maniera qualitativa si può dedurre che questa caratteristica è pienamente soddisfatta con la semplice rotazione delle zampe tramite un unico attuatore. L'implementazione di un'ulteriore attuatore permetterebbe il controllo indipendente di ciascuna zampa, aggiungendo un grado di libertà, ovvero l'altezza del piano di appoggio, ma complicando notevolmente il progetto, perchè implicherebbe uno studio sofisticato sulla dinamica dei servomotori. Si reputa quindi sufficiente l'impiego di un solo attuatore, il quale muoverà contem-

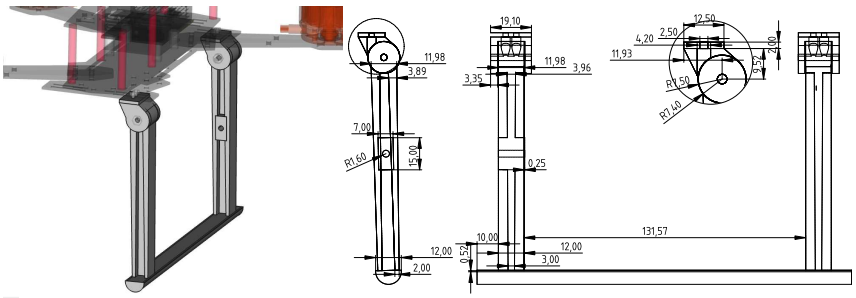

Figura 23: Rendering 3D di una delle zampe applicato al QAV250 (a sinistra) e relativo disegno tecnico (a destra)

poraneamente entrambe le zampe tramite un apposito meccanismo articolato.

3.2 ARCHITETTURA DELLA STRUTTURA

3.2.1 *Geometria delle zampe*

La progettazione della forma delle zampe è svolta a partire dal disegno 3D del drone, cercando di sfruttare la struttura presistente e i relativi fori fin'ora inutilizzati. Si è scelto l'impiego di due perni per zampa in modo da rendere più robusta la struttura di atterraggio minimizzando contemporanemente il peso complessivo delle zampe (Figura 23). Questo sistema, infatti, dovrà garantire solidità in condizioni di impatto anche brusco e in situazioni limite in cui una delle zampe si trovi ad essere in posizione verticale, considerabile quindi come una trave a sbalzo. I perni di una stessa gamba sono quindi posizionati più distanti possibile uno dall'altro, il che implica una modifica della piastra sottostante del drone, la quale risulta troppo piccola per ospitare i perni nella configurazione ad assi distinti con $b = 0.050\ m$. Si sceglie di partire dalla geometria della piastra di fabbrica riportando gli stessi fori e andando ad aggiungere delle superfici su cui fissare i perni (Figura 24). Si è deciso inoltre di creare dei fori asolati per alloggiare i bulloni che fisseranno i perni alla piastra, così da poterne in seguito modificare la distanza b. Il perno presenta due occhielli i quali ospiteranno il bullone che fissa la zampa e ne permette la rotazione. Questi sono di grandi dimensioni per poter sopportare le forze che possono crearsi in casi estremi definiti poco fa. Il resto della geometria è disegnata in maniera da essere quasi complementare alla geometria di collegamento della zampa, ma lasciando un leggero gioco per evitare lo sfregamento tra le superfici e minimizzando quindi la forza di attrito che potrebbe venire a crearsi durante la rotazione. Per la progettazione delle aste delle zampe, per massimizzare la resistenza a rottura, si è scelto l'impiego di una sezione ad H, la quale permette di ottenere un'ottima resistenza a flessione mi-

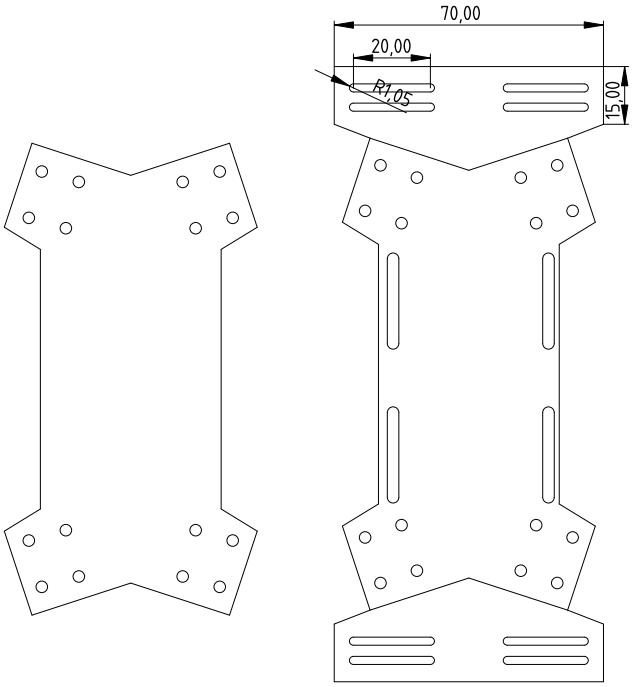

Figura 24: Disegno della piastra del QAV250 originale (a sinistra) e della versione modificata (a destra)

nimizzando la quantità di materiale necessario. Inoltre, essendo che la zona più sollecitata a flessione è quella in prossimità del perno, si è aumentata la dimensione della sezione proprio nei pressi di questo punto, andando a ridurla in maniera lineare man mano che ci si avvicina al pattino.

3.2.2 *Studio del meccanismo*

Scegliendo di impiegare un unico azionamento per la movimentazione contemporanea di entrambe le zampe in maniera dipendente una dall'altra, si studia quindi un meccanismo articolato in grado di adempiere a questo compito. Questo andrà a collegarsi, da una parte, a ciascuna delle zampe e, dall'altra, al perno del servomotore. Per la risoluzione di questo problema vengono imposte a priori le seguenti ipotesi:

- la distanza tra i perni delle zampe è nota (derivante dallo studio di cui sottosezione 3.1.4) e pari a $b = 0.050\ m$;
- per questioni di simmetria, l'asse di rotazione del servomotore è centrato rispetto alla struttura del drone;
- la posizione del servomotore è fissa e dipendente dalla batteria;
- il servomotore può imprimere una rotazione compresa tra $-90°$ e $+90°$;

- il meccanismo nel suo complesso deve essere simmetrico.

Si ipotizza inoltre che, per riuscire a sopperire alla condizione di inclinazione massima del piano di appoggio, il meccanismo venga studiato proprio a partire da una delle sue configurazioni limite, ovvero corrispondente ad avere una zampa ruotata in posizione orizzontale verso l'esterno e l'altra in posizione verticale. Una volta definito il meccanismo che permette di ottenere questa configurazione, la condizione limite inversa è automaticamente soddisfatta per simmetria. Si impone, inoltre, che in ciascuna di queste condizioni il perno rotante del servomotore si trovi anch'esso in una delle sue configurazioni limite. Ciò comporta che, quando le zampe saranno nella condizione di piano di appoggio di inclinazione massima, il servomotore si troverà ad essere ruotato di $-90°$ e sarà invece a $+90°$ nella condizione inversa. A partire da queste ipotesi, i parametri incogniti del problema si riducono a (vedi Figura 25):

- x: raggio del perno del servomotore;
- y: lunghezza dei bracci del meccanismo;
- z: lunghezza del tratto compreso tra l'asse di rotazione della zampa ed il punto di collegamento con il meccanismo.

Nominando i vettori come in figura 26, ovvero:

- u_a: vettore tra il punto di rotazione della zampa sinistra e il relativo punto di fissaggio del meccanismo alla zampa stessa;
- u_b: vettore che corrisponde al tratto di meccanismo che collega la zampa sinistra al servomotore;
- u_c: vettore che corrisponde al tratto di meccanismo che collega la zampa destra al servomotore;
- u_d: vettore che va dal punto di rotazione del servomotore al punto di collegamento dei due meccanismi;
- u_e: vettore tra il punto di rotazione della zampa destra e il relativo punto di fissaggio del meccanismo alla zampa stessa.

Riportandosi ad uno dei casi limite, possiamo affermare che:

$$\begin{cases} x_{u_a} = -z \\ x_{u_d} = -x \\ y_{u_e} = -z \end{cases} \tag{10}$$

Tramite delle operazioni tra vettori e sostituendo le equazioni (10) possiamo ricavare:

$$-x_{u_a} + \frac{b}{2} = -x_{u_d} + x_{u_b} \Rightarrow z + \frac{b}{2} = x + x_{u_b} \tag{11}$$

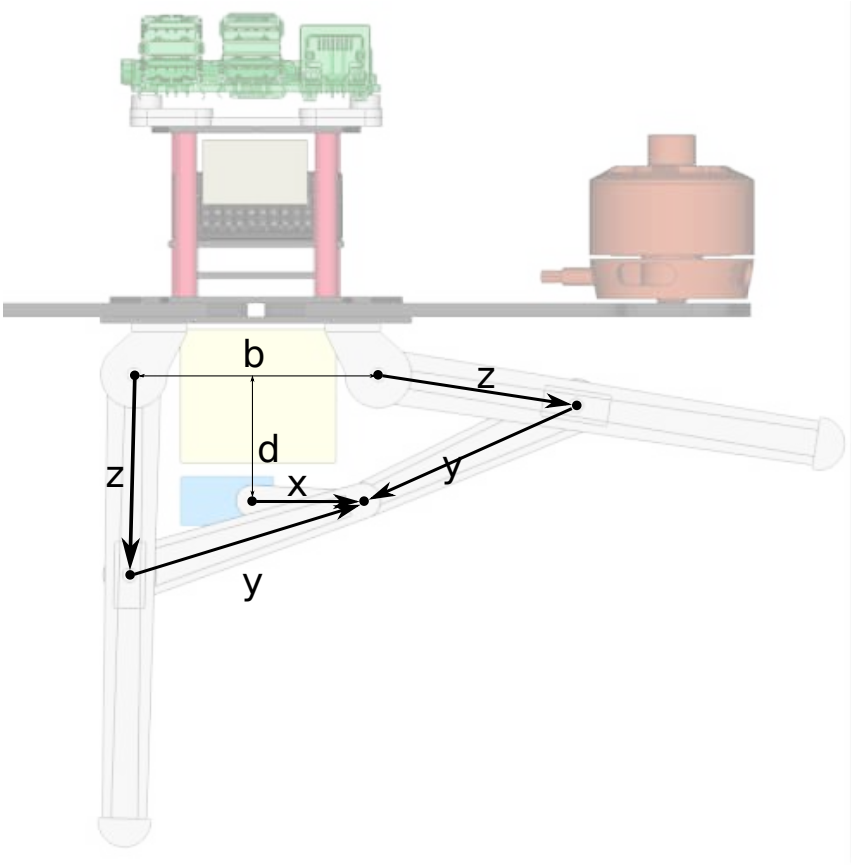

Figura 25: Disegno bozza delle zampe e del meccanismo con rappresentazione dei relativi vettori

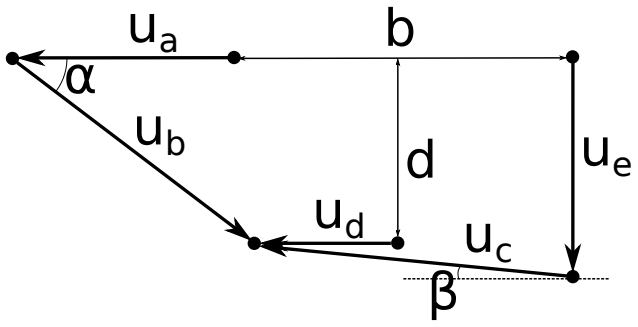

Figura 26: Rappresentazione dei vettori del meccanismo e delle zampe in una delle configurazioni limite

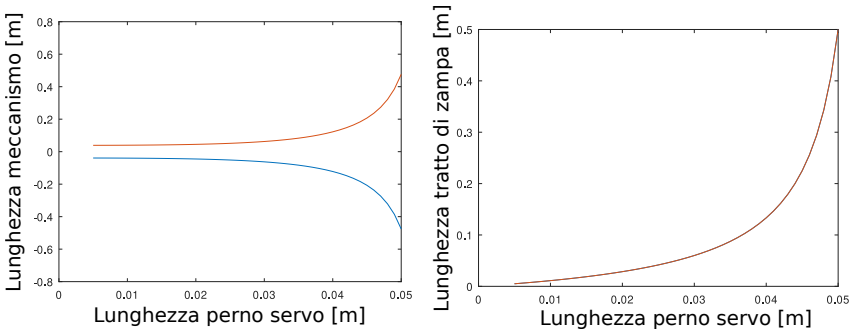

Figura 27: Plot dei valori della lunghezza del meccanismo (y) (a sinistra) e della lunghezza del tratto di zampa (z) al variare del raggio del perno del servomotore (x)

$$d = -y_{u_b} \tag{12}$$

$$\frac{b}{2} = -x_{u_c} + x_{u_d} \Rightarrow \frac{b}{2} = -x_{u_c} - x \tag{13}$$

$$-y_{u_e} = y_{u_c} + d \Rightarrow z = y_{u_c} + d \tag{14}$$

Definendo gli angoli α e β come in figura vale:

$$\begin{cases} x_{u_b} & = y\cos(\alpha) \\ -y_{u_b} & = y\sin(\alpha) \\ -x_{u_c} & = y\cos(\beta) \\ y_{u_c} & = y\sin(\beta) \end{cases} \tag{15}$$

Sostituendo alle equazioni (11)-(14) e raggruppandole otteniamo:

$$\begin{cases} z + \frac{b}{2} = x + y\cos(\alpha) \\ d = y\sin(\alpha) \\ \frac{b}{2} = y\cos(\beta) - x \\ z = y\sin(\beta) + d \end{cases} \tag{16}$$

Il sistema (16) è composto da 4 equazioni e presenta 5 incognite: x, y, z, α, β. La sua risoluzione, quindi, sarà un sistema di equazioni dipendenti da un'unica incognita. Questo implica che non esiste un'unica possibile configurazioni ma si può scegliere tra infinite soluzioni possibili. Una volta definiti i valori noti, la risoluzione viene svolta tramite il software GNU Octave, utilizzando il seguente codice:

```
syms x y z alfa beta;
eqns = [z + b/2 == x + y*cosd(alfa), ...
d == y*sind(alfa), ...
b/2 == y*cosd(beta) - x, ...
z == y*sind(beta) + d ];
S = solve (eqns, [alfa, beta, y, z]);
```

A partire dai grafici in figura 27, rappresentanti i valori di y e z al variare di x, si può scegliere qualitativamente dei valori che siano geometricamente accettabili, in maniera che le dimensioni non siano troppo piccole (con conseguenti forze in gioco nei perni troppo elevate) e nemmeno troppo grandi (ingombro e peso eccessivo). In entrambi i grafici si possono notare la presenza di due soluzioni distinte del sistema, a sinistra uguali ma di segno opposto e a destra coincidenti. Tuttavia, non essendo fisicamente accettabili lunghezze di valore negativo, si considera solo la soluzione arancione ovvero quella che presenta valori di y positivi. I valori scelti, quindi, sono:

$$x = 0.025\ m \qquad y = 0.0514\ m \qquad z = 0.0385\ m$$

3.3 CINEMATICA PER IL SISTEMA DI ATTERRAGGIO

Una volta scelta la configurazione del meccanismo, risulta necessario un doppio studio della sua cinematica:

- Cinematica diretta: permette di definire, a partire dall'angolazione del perno del servomotore, la rotazione delle zampe e, di conseguenza, anche l'orientazione del piano di contatto dei pattini. Viene impiegata in fase di simulazione per movimentare le zampe a partire dal dato di posizione del servomotore.
- Cinematica inversa: ricostruisce la rotazione che dovrebbe avere il servomotore a partire dall'orientazione delle zampe e, successivamente, a partire dall'inclinazione della superficie di atterraggio. Questo studio viene implementato nel controllore dell'azionamento per comandare il servo.

Nel capitolo successivo se ne approfondiranno le implementazioni specifiche all'interno del codice di programmazione. Ci si limita ora, quindi, al solo studio analitico tramite l'ausilio di GNU Octave.

3.3.1 *Cinematica diretta del meccanismo*

Riportandoci ancora una volta ad una semplificazione bidimensionale (Figura 28), sono noti:

- γ: rotazione del perno del servomotore rispetto all'asse verticale, che sarà il parametro del problema;

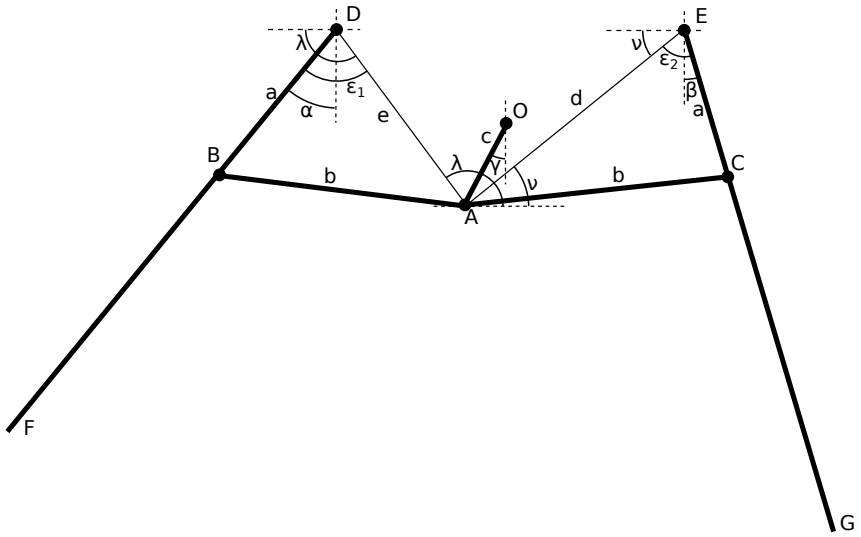

Figura 28: Schematizzazione bidimensionale della cinematica diretta

- x_D, y_D, x_E, y_E: le coordinate degli assi di rotazione delle due zampe, derivanti dagli studi precedenti;
- a: lunghezza dei tratti di zampa compresi tra l'asse di rotazione e l'attacco al meccanismo;
- b: lunghezza dei bracci del meccanismo;
- c: lunghezza del perno del servomotore;
- l: lunghezza delle zampe.

Fissando il centro del perno del servomotore come origine del sistema di riferimento ($O = (0;0)$), si possono trovare le coordinate del punto A:

$$x_A = c\sin(\gamma) \tag{17}$$
$$y_A = x\cos(\gamma) \tag{18}$$

Partendo dalla zampa sinistra, si ricava la lunghezza del tratto e, ovvero la distanza del segmento compreso tra A e D, a partire dalle loro coordinate cartesiane:

$$e = \sqrt{(y_D - y_A)^2 + (x_E - x_A)^2} \tag{19}$$

Ora si può utilizzare il teorema del coseno per affermare:

$$b^2 = a^2 + e^2 - 2ae\cos(\epsilon_1) \tag{20}$$

dalla quale ricavare il valore di ϵ_1

$$\epsilon_1 = \arccos\left(\frac{a^2 + e^2 - b^2}{2ae}\right) \tag{21}$$

A partire dalle coordinate dei punti A e D si potrebbe ricavare l'angolo λ tramite la funzione arc tan correggendo eventualmente di $\pm\pi$. Si preferisce, tuttavia, utilizzare la funzione arc tan2 [1], già implementata nella maggior parte dei linguaggi di programmazione:

$$\lambda = \text{arc tan2}(y_D - y_A, x_D - x_A) \tag{22}$$

E' inoltre vera la seguente operazione tra angoli:

$$\alpha = 90^\circ - \lambda + \epsilon_1 \tag{23}$$

Dove α è l'angolo che d'ora in poi descriverà la rotazione della zampa sinistra.

Analogamente, possiamo effettuare le stesse operazioni anche sulla zampa destra, brevemente:

$$b^2 = a^2 + d^2 - 2ad\cos(\epsilon_2) \tag{24}$$

$$\epsilon_2 = \arccos\left(\frac{a^2 + d^2 - b^2}{2ad}\right) \tag{25}$$

$$\nu = \text{arc tan2}(y_E - y_A, x_E - y_A) \tag{26}$$

$$\beta = \epsilon_2 - 90^\circ + \nu \tag{27}$$

Avendo a disposizione, ora, i valori di α e β, possiamo ricavare le coordinate dei punti corrispondenti ai pattini:

$$\begin{cases} x_G = x_E + l\sin(\beta) \\ y_G = y_E - l\cos(\beta) \\ x_F = x_D - l\sin(\alpha) \\ y_F = y_D - l\cos(\alpha) \end{cases} \tag{28}$$

L'inclinazione del piano di atterraggio sarà:

$$\theta = \text{arc tan2}(y_G - y_F, x_G - x_F) \tag{29}$$

3.3.2 *Approssimazione della cinematica diretta del meccanismo*

Nel precedente paragrafo (3.3.1) si sono ricavati i valori di α, β e θ (rispettivamente rotazione zampa sinistra, rotazione zampa destra e

1 L'arcotangente in questa applicazione viene utilizzata per ricavare l'angolo c compreso tra l'asse x e un generico punto di coordinate (x, y), con centro nell'origine del piano cartesiano. Con arc tan2(x, y) si intende una variante di arc tan$(\frac{x}{y})$ la quale permette di distinguere anche i segni del punto, identificando in quale quadrante si trova e quindi fornendo un risultato nel dominio $(-\pi, \pi)$ senza necessitare di correzione dell'angolo a posteriori.

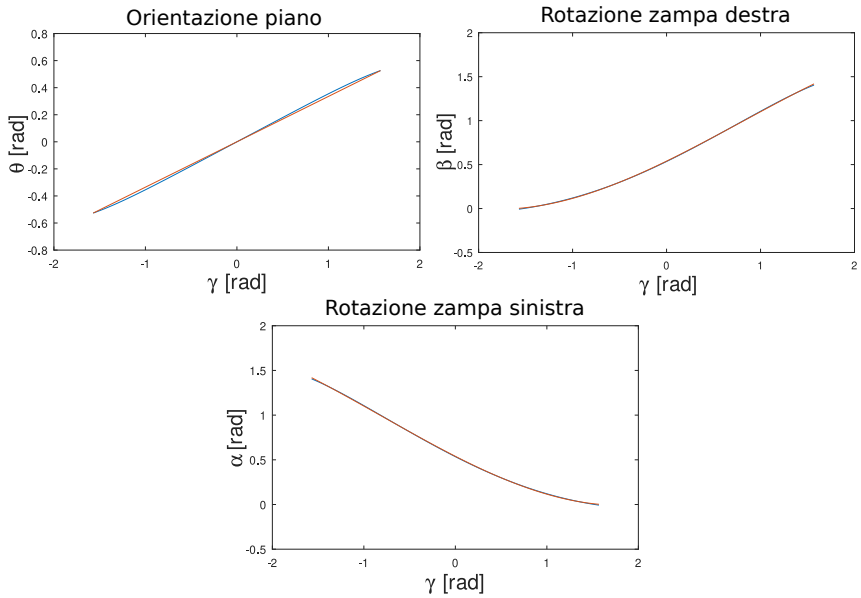

Figura 29: Plot dei valori di θ, α e β al variare di γ a partire dalle formule analitiche (curva blu) e relativa approssimazione (curva arancione)

inclinazione del piano) a partire dal parametro γ (rotazione del perno del servomotore) e sfruttando concetti della trigonometria. Tuttavia, per riuscire ad implementare il tutto all'interno del simulatore, si sceglie di semplificare la quantità di calcoli richiesti optando per un'approssimazione di tipo polinomiale da eseguire una tantum. In questa maniera si vuole ricavare una funzione polinomiale di grado opportuno, dipendente da γ, per ognuno dei valori sopracitati. Appoggiandosi a GNU Octave si procede a calcolare i valori di α, β e θ, utilizzando le formule presentate in (3.3.1), per vari valori di γ compreso tra $-\pi/2$ e $\pi/2$ (limiti massimi di rotazione del servomotore) e archiviandoli rispettivamente all'interno dei vettori *alfa*, *beta* e *ang*. Scegliendo un grado del polinomio sufficientemente elevato da approssimare correttamente i valori ottenuti, è possibile calcolare le relative interpolazioni come segue:

```
coef1 = polyfit(gamma, alfa, 3);
coef2 = polyfit(gamma, beta, 3);
coef3 = polyfit(gamma, ang, 1);
```

i cui risultati sono rappresentati in figura 29. Dall'ultima riga di codice è utile osservare che il legame tra γ e θ, nonostante la presenza del meccanismo articolato, è quasi lineare o, almeno, approssimabile come tale. Per α e β, invece, si riesce a raggiungere un buon risultato con un polinomio del terzo grado.

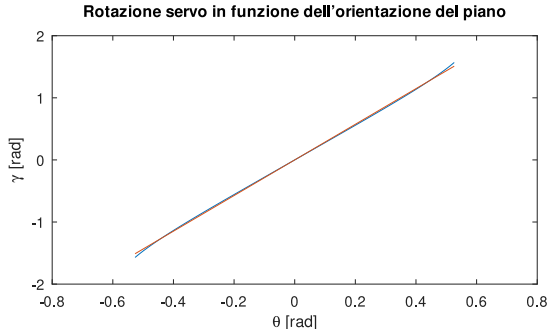

Figura 30: Plot dei valori di γ al variare di θ a partire dalle formule analitiche (curva blu) e relativa approssimazione (curva arancione)

3.3.3 *Approssimazione cinematica inversa del meccanismo*

Nella sezione 3.3.1 si è ricavato la formulazione analitica che lega l'inclinazione del piano d'atterraggio con il valore di γ e considerando significativa soltanto una delle due soluzioni. Avendo già a disposizione una funzione espressa sotto forma di punti discreti, calcolati tramite GNU Octave, è possibile svolgere nuovamente un'interpolazione polinomiale per ricavarci la cinematica inversa del meccanismo. Quindi, a partire dai vettori già creati precedentemente, si ricava la funzione inversa del legame tra γ e θ

```
coef4 = polyfit(ang, gamma, 1);
```

Trattandosi di una funzione inversa il risultato, visibile in Figura 30, risulta quindi il medesimo di Figura 29 in alto a sinistra con gli assi cartesiani invertiti.

4

SISTEMA DI ATTERRAGGIO: ANALISI DELLE PRESTAZIONI IN AMBIENTE VIRTUALE

Prima di andare ad implementare fisicamente tutto ciò che si è andati a progettare nel corso di questa trattazione, si opta per una prima fase di simulazione di ciò che si è realizzato. Gli aspetti che hanno portato a questa scelta possono essere schematizzati come segue:

- Money-saving: testare la soluzione in ambiente simulativo consente di non dover disporre di aree adibite al volo in sicurezza, quali gabbie indoor o ampie aree all'aperto, e di evitare danni al drone a causa di comportamenti inaspettati dovuti ad errori nella prototipazione.
- Diagnostica: una volta configurata la simulazione e preparato il modello, si possono ripetere le prove con estrema facilità, ripristinando l'ambiente alle condizioni iniziali ogni volta. Inoltre, grazie all'interfacciamento con il motore fisico, si è in grado di ricavare facilmente misure che nel drone reale richiederebbero sofisticate apparecchiature.
- Long-term: una volta completata la preparazione del simulatore, si permette uno sviluppo più agile per le persone che ne proseguiranno la progettazione.

Si riportano tuttavia delle motivazione sfavorevoli all'uso del simulatore:

- l'interazione tra gli elementi simulati è molto complessa da configurare in maniera congruente alla realtà. Si creano molto spesso degli artifizi difficilmente giustificabili e che a volte richiedono molto tempo per essere risolti;
- la preparazione del modello è molto esosa in termini di tempo;
- gli strumenti software che si utilizzano sono ancora in fase di sviluppo e quindi spesso sono poco user-friendly.

4.1 PREPARAZIONE DEL MODELLO SIMULATIVO

4.1.1 *Modello CAD*

Per facilitare la progettazione dei pezzi che verranno aggiunti alla struttura del drone o per eventuali modifiche di quelli già esistenti all'interno del kit è necessario avere a disposizione un disegno CAD

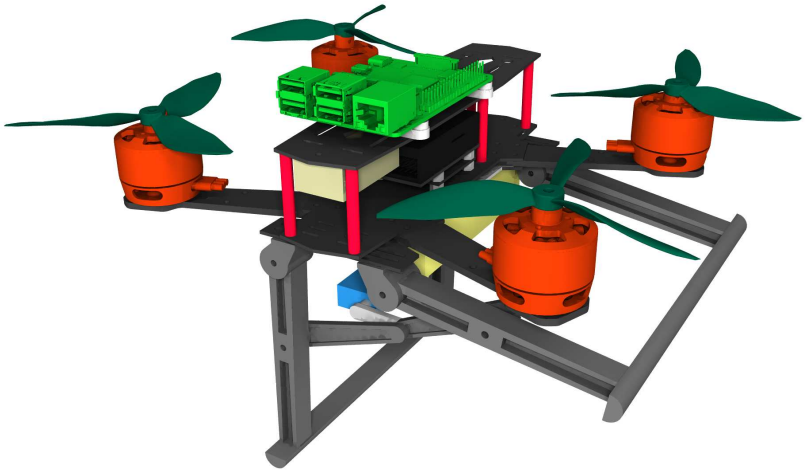

Figura 31: Rendering del progetto CAD del QAV250

dettagliato dell'intero velivolo. Inoltre, avendo a disposizione un modello 3D, la creazione dei file mesh impiegati nella simulazione e la procedura per ricavare le inerzie risulteranno immediate. Non essendo presenti in rete questo tipo di file relativi alla versione del QAV250 in uso, si è optato per la creazione da zero dell'intero modello a partire dalle misurazioni delle dimensioni dei singoli pezzi (Figura 31). Come programma per il disegno 3D si è impiegato FreeCAD, un software open-source specializzato nel disegno parametrico generico sia in ambiente bidimensionale che tridimensionale. Tutti i pezzi del drone sono stati smontati dal drone per essere misurati uno ad uno tramite l'ausilio di un calibro, ad esclusione della Pixhawk, dei motori e degli ESC, i quali sono stati ricavati da modelli presenti in rete e già sufficientemente dettagliati per i nostri scopi. Eventuali fori e fessure presenti sulle superfici in fibra di carbonio sono stati riportarti solo nei casi in cui risultassero significativi per il calcolo delle inerzie o potessero risultare utili in seguito per il fissaggio di altri accessori. Sono state inoltre escluse da questa procedura eventuali viterie e bullonerie.

4.1.2 *Dinamica dei componenti*

L'implementazione dell'intero drone all'interno di un simulatore fisico richiede la conoscenza delle caratteristiche dei componenti, quali massa, centro di massa, inerzia e caratterizzazione delle interazioni, sia col mondo esterno, sia intrinsecamente. Questi valori devono essere sufficientemente accurati allo scopo di ottenere dei risultati realisti e significativi per lo studio. Una delle caratterizzazioni più importanti riguarda la stima delle masse e delle inerzie dei singoli pezzi che compongono il drone, sia quelli già presenti di fabbrica, sia l'hardware aggiunto a posteriori. Tutti gli altri parametri, invece, verranno affrontati in seguito, specificatamente per il simulatore in uso.

La massa di tutti i componenti presenti a bordo del drone quali piastre, motori, Pixhawk, ecc. è misurata singolarmente smontando la struttura nei singoli pezzi e servendosi di una bilancia sufficientemente precisa (incertezza di ± 1 g). I pezzi, invece, che non sono ancora stati implementati fisicamente sul drone quali zampe e relativi supporti, essendo costruiti tramite stampa 3D, hanno una massa facilmente stimabile importando il loro modello nel software di *slicing* e configurando la stampa a dovere. Ad ogni modo, per semplicità, si escludono da questo processo tutti quei componenti con bassa inerzia che non sono particolarmente influenti nei calcoli e che spesso richiedono una trattazione troppo complicata per loro natura, come nel caso dei vari cavi di collegamento che possono assumere forme del tutto casuali e possono muoversi col tempo. A partire dalle masse, conoscendo la geometria e la disposizione dello spazio di ciascun pezzo, è possibile calcolare i momenti di inerzia. Questa operazione, essendo troppo complicata per essere svolta analiticamente, viene effettuata appoggiandosi ad una *Macro* presente all'interno di FreeCAD, nominata *FCInfo* [14], la quale consente di ricavare informazioni sull'inerzia e sul centro di massa semplicemente selezionando il relativo assieme. Si ipotizza, tuttavia, che ciascun singolo pezzo sia considerabile come composto da un unico materiale omogeneo, che comunque risulta un'ottima approssimazione se si considera componente per componente come in questo caso.
Per quanto riguarda il momento d'inerzia, i dati saranno nella forma di una matrice $3x3$ nominata tensore d'inerzia del singolo componente rispetto al suo stesso centro di massa, che viene riportata in tabella 1 riga per riga, seguendo la seguente rappresentazione:

$$I_{cm} = \begin{bmatrix} I_{cm,xx} & I_{cm,xy} & I_{cm,xz} \\ * & I_{cm,yy} & I_{cm,yz} \\ * & * & I_{cm,zz} \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} \tag{30}$$

dove I_cm è una matrice simmetrica e

- $I_{cm,xx}$, $I_{cm,yy}$ e $I_{cm,zz}$ sono i momenti d'inerzia rispetto agli assi x, y e z
- $I_{cm,xy}$, $I_{cm,xz}$ e $I_{cm,yz}$ sono i prodotti d'inerzia

Infine, il centro di massa in tabella 1 è espresso rispetto all'origine degli assi cartesiani del modello 3D, il quale è stato scelto arbitrariamente a inizio progettazione e non corrisponde a nessun punto significativo.

	Peso g	Densità kg/dm^3	I_1 $g\ mm^2$	I_2 $g\ mm^2$	I_3 $g\ mm^2$	C_m mm
Terza piastra	13	1.41	34516.5366 -0.0 12.3838	-0.0 36996.1755 -0.0	12.3838 -0.0 2484.5254	0.0191 31.766 100.4606
Ingombro batteria	211	2.01	191703.75 0.0 0.0	0.0 197419.6875 0.0	0.0 0.0 37373.4375	0.5 -28.0 93.0
Supporto Raspberry	8	1	2189.1376 -0.0 -0.0003	-0.0 3117.7271 -0.0	-0.0003 -0.0 945.3144	0.0 34.7285 117.0
Motore 1	34	1.7	877.8745 -0.0 -0.003	-0.0 1010.3867 -0.0	-0.003 -0.0 877.8693	93.4111 1.2357 175.0513
Motore 2	34	1.7	877.8745 0.0 0.003	0.0 1010.3867 -0.0	0.003 -0.0 877.8693	-93.4111 1.2357 175.0513
Motore 3	34	1.7	877.8745 -0.0 0.003	-0.0 1010.3867 0.0	0.003 0.0 877.8693	93.4111 1.2357 24.9487
Motore 4	34	1.7	877.8745 -0.0 -0.003	-0.0 1010.3867 -0.0	-0.003 -0.0 877.8693	-93.4111 1.2357 24.9487
Raspberry	50	2.68	12081.3494 0.0 -26.4133	0.0 17237.6239 0.0	-26.4133 0.0 5164.8534	-0.1726 44.0 127.1208
Ricevitore RC	11	0.76	1849.5044 -0.0 0.0	-0.0 2163.1818 -0.0	0.0 -0.0 699.3539	-0.25 22.266 162.25
Base	17.24	1.41	40466.4201 -0.0 -0.3201	-0.0 45051.3381 0.0	-0.3201 0.0 4591.3814	-0.0179 -10.934 100.0359
Ala 1	9.25	1.41	1625.4246 0.0 -3234.8613	0.0 10316.3322 0.0	-3234.8613 0.0 8704.7851	56.7313 -8.684 160.9679
Ala 2	9.25	1.41	1625.4246 -0.0 3234.8613	-0.0 10316.3322 0.0	3234.8613 0.0 8704.7851	-56.7313 -8.684 160.9679
Ala 3	9.25	1.41	1625.4246 0.0 3234.8613	0.0 10316.3322 -0.0	3234.8613 -0.0 8704.7851	56.7313 -8.684 39.0321
Ala 4	9.25	1.41	1625.4246 0.0 -3234.8613	0.0 10316.3322 -0.0	-3234.8613 -0.0 8704.7851	-56.7313 -8.684 39.0321
Supporto 1	1.2	1	62.1245 -3.8283 -0.0	-3.8283 55.9197 -0.0	-0.0 -0.0 34.3728	-23.8977 -17.495 171.9144
Supporto 2	1.2	1	62.1245 3.8283 0.0	3.8283 55.9197 -0.0	0.0 -0.0 34.3728	23.8977 -17.495 171.9144
Supporto 3	1.2	1	62.1245 3.8283 -0.0	3.8283 55.9197 0.0	-0.0 0.0 34.3728	23.8977 -17.495 28.0856
Supporto 4	1.2	1	62.1245 -3.8283 0.0	-3.8283 55.9197 0.0	0.0 0.0 34.3728	-23.8977 -17.495 28.0856
Pixhawk	37.2	1.3	8808.1026 -20.5268 2.9668	-20.5268 12465.8292 -110.1511	2.9668 -110.1511 5251.1747	0.3831 14.5708 108.5679
Seconda piastra	17		41216.168 -0.0 0.0	-0.0 44840.135 0.0	0.0 0.0 3630.5131	-0.0 -6.434 99.8437

Tabella 1: Tabella dei valori di massa, momento di inerzia e centro di massa calcolati a partire dal modello 3D

A seguito di prove svoltesi tramite il simulatore Gazebo, si evince che l'iterazione tra elementi considerabili collegati in maniera fissa uno con l'altro risulta essere problematica, originando perturbazioni nell'intera struttura. Ciò è causato principalmente dal modello equivalente molla-smorzatore che lega ciascun pezzo e sul quale Gazebo si basa per il calcolo delle forze. Si opta dunque all'utilizzo di un modello equivalente, composto dal minor numero di componenti possibili, che si ottiene raggruppando tutti quei pezzi che possiamo considerare fissi uno rispetto all'altro, ovvero tutti quelli raggruppati in tabella 1. Va quindi ricavato un'unico tensore d'inerzia rispetto all'origine, il quale può essere calcolato come la somma dei tensori d'inerzia dei singoli componenti sempre rispetto all'origine, ovvero:

$$I_O^{TOT} = \sum_{k=1}^{N} I_O^k \tag{31}$$

Dove I_O^k può essere calcolato a partire dal tensore di inerzia del singolo componente rispetto al suo centro di massa sfruttando il teorema di Huygens-Steiner nel caso di tensori, il quale enuncia in formula:

$$I_O^k = I_{cm}^k + m_k S(P_{cm}^k)^T S(P_{cm}^k) \tag{32}$$

con:

- I_{cm}^k è il tensore d'inerzia del k-esimo elemento rispetto al suo centro di massa
- m_k è la massa del k-esimo elemento
- $S(P_{cm}^k)$ è la matrice antisimmetrica costruita a partire dal centro di massa come segue:

$$S(P_{cm}^k) = \begin{bmatrix} 0 & -z_{cm}^k & y_{cm}^k \\ z_{cm}^k & 0 & -x_{cm}^k \\ -y_{cm}^k & x_{cm}^k & 0 \end{bmatrix} \tag{33}$$

  dove il centro di massa è definito come $P_{cm}^k = \begin{bmatrix} x_{cm}^k & y_{cm}^k & z_{cm}^k \end{bmatrix}$

Il centro di massa globale invece sarà dato dalla media ponderata sulla massa dei singoli centri di massa:

$$P_{cm}^{TOT} = \frac{1}{M} \sum_{k=1}^{N} P_{cm}^k \, m_k \tag{34}$$

con massa totale

$$M = \sum_{k=1}^{N} m_k \tag{35}$$

[20]

I valori che si ottengono sono i seguenti:

$$M = 483\ g \qquad I_O^{TOT} = \begin{bmatrix} 6557600 & 3384 & -8277 \\ 3384 & 7622400 & 190230 \\ -8277 & 190230 & 1690900 \end{bmatrix} g\ mm^2 \quad (36)$$

4.2 SIMULAZIONE

Prima di andare ad implementare fisicamente tutto ciò che si è andati a progettare nel corso di questa trattazione, si opta per una prima fase di simulazione di ciò che si è realizzato. Gli aspetti che hanno portato a questa scelta possono essere elencati come segue:

- non richiedere strutture adibite al volo in sicurezza, quali gabbie indoor o ampie aree sgombre all'aperto;
- durante la prototipazione è facile commettere degli errori e produrre dei bug, i quali potrebbero diventare pericolosi per gli operatori nel caso di implementazione direttamente sul drone;
- non si rischia di far impattare il drone, provocandone la rottura o la deformazione;
- è possibile sviluppare il progetto senza doversi recare in laboratorio;
- una volta configurata la simulazione e preparato il modello, si possono ripetere le prove con estrema facilità, ripristinando l'ambiente alle condizioni iniziali ogni volta;
- non richiede il collegamento di cavi a schermi, computer e altri strumenti di diagnostica. Non è necessario attendere il riavvio di tutti i dispositivi a seguito di cambiamenti, anche minimi, delle configurazioni. Non è necessario sostituire le batteria con altre cariche;
- Una volta preparato il progetto dal punto di vista simulativo, si permette uno sviluppo più agile per le persone che ne proseguiranno la progettazione.

Si riportano tuttavia delle motivazione sfavorevoli all'uso del simulatore:

- l'interazione tra gli elementi simulati è molto complessa da configurare in maniera congruente alla realtà. Si creano molto spesso degli artifizi difficilmente giustificabili e che a volte richiedono molto tempo per essere risolti;
- la preparazione del modello è molto esosa in termini di tempo;

- gli strumenti software che si utilizzano sono ancora in fase di sviluppo e quindi spesso sono poco user-friendly.

4.2.1 *Gazebo*

Gazebo 11 è una software simulativo open-source, impiegato principalmente nel campo robotico, il quale presenta la possibilità di renderizzare graficamente l'ambiente, creando dei modelli sia dell'ambiente sia del modello del robot, in maniera molto dettagliata. Il comportamento dinamico e l'interazione tra elementi sono calcolati scegliendo tra uno dei motori fisici generici disponibili, anch'essi tutti open-source, tra i quali:

- *ODE* (*Open Dynamics Engine*): è il più famoso tra i motori fisici ed è quello che Gazebo utilizza di default. Il suo sviluppo è ad uno stadio oramai avanzato, tanto che nella storia è stato impiegato anche all'interno di alcuni giochi per computer.
- *Bullet*: è uno dei motori fisici più veloci e questo lo rende molto accurato in quanto, a parità di capacità computazionale, permette di eseguire più interazioni e con incertezze minori. Tuttavia, la gestione delle interazioni tra elementi collegati è di tipo esplicito[1], caratteristica che potrebbe comportare un aumento dei fenomeni di instabilità.
- *Simbody*: sviluppato principalmente come simulatore nell'ambito della biomeccanica, è in grado di gestire anche geometrie complesse. E' in generale tra i più affidabili.
- *DART* (*Dynamic Animation and Robotics Toolkit*): molto simile ad ODE come attendibilità, ma utilizza una gestione delle iterazioni di tipo implicito. Supporta anche geometrie complesse. [5]

Per la simulazione del modello del drone verrà utilizzato principalmente il primo tra questi in quanto risulta essere il più compatibile per l'iterazione col firmware di PX4. Tuttavia, nel corso delle sperimentazioni, sono state provate anche le altre soluzioni al fine di cercare di incrementare la stabilità del sistema.

La renderizzazione grafica invece è affidata ad OpenGL, una API molto impiegata in ambito Linux per la creazione di GUI e grafiche 3D, molto versatile in quanto compatibile con molte piattaforme e utilizzabile con diversi codici di programmazione. La simulazione è composta da due sotto-programmi, uno dei quali denominato server,

1 Si dice di tipo esplicito un'interazione in cui la velocità al tempo $t+1$ è calcolata con i dati di posizione e velocità calcolati al tempo t. Questa tecnica è considerata più semplice da computare. E' di tipo implicito, invece, quando vengono usati dati di posizione e velocità al tempo $t+1$.

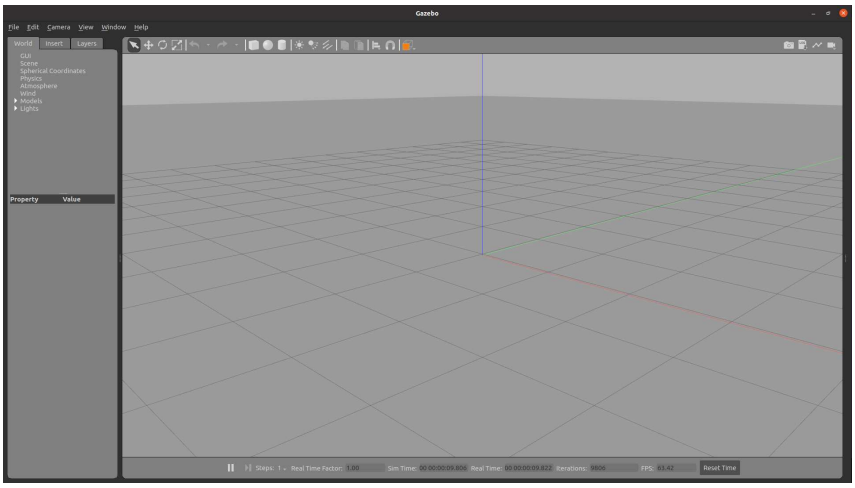

Figura 32: Esempio di schermata iniziale di Gazebo Client

il quale si occupa di effettuare i calcoli veri e propri, e un secondo, nominato client (Figura 32), che invece si occupa del rendering grafico e dell'interfacciamento con l'utente. Questi comunicano tra di loro tramite una connessione TCP, tipicamente localhost, ma ipoteticamente potrebbero essere eseguiti su due computer diversi appartenenti alla stessa rete locale. Per eseguirli, è necessario inserire i seguenti comandi in due terminali distinti:

```
$ gzserver <nome_file_sdf>
$ gzclient
```

La modellizzazione dell'ambiente (*world*) e degli oggetti specifici (*model*) avviene tramite il linguaggio SDF (Simulation Description Format), ovvero un codice di tipo XML appositamente creato per Gazebo e quindi ottimizzato per la simulazione robotica. La progettazione di modelli in SDF può essere effettuata, nel caso di sistemi poco complessi, tramite il model editor presente all'interno del client di Gazebo, oppure editando manualmente il file. Gli elementi base che si possono inserire all'interno del tag principale (<*sdf*>) sono:

- *world*: permette di definire l'ambiente, la superficie, la visualizzazione grafica, il motore fisico e tutte quelle configurazioni globali che interessano tutti gli elementi presenti nella simulazione.
- *scene*: descrive l'aspetto dell'ambiente, come ad esempio l'illuminazione presente ed il cielo.
- *state*: è lo stato che tutti gli elementi presenti devono assumere ad uno specifico istante (ad esempio le loro posizioni iniziali).
- *physics*: specifica il motore fisico da utilizzare e tutti i parametri richiesti per il calcolo della simulazione.

- *light*: per impostare le sorgenti luminose utilizzate per il rendering.
- *model*: contiene gli elementi che descrivono il sistema che si vuole simulare ad esempio un robot o, nel nostro caso, il drone.
- *link*: ciascun singolo elemento che compone il model e che interagisce con gli altri link.
- *joint*: legame di interazione tra un link genitore ed uno figlio, il quale può limitarne i gradi di libertà nel movimento relativo tra i due o l'ampiezza del movimento stesso. Un joint può essere ad esempio un legame a cerniera, oppure un collegamento di tipo fisso.
- *sensor*: associabile sia ad un joint che ad un link, ne fornisce la possibilità di acquisizione di dati sullo stato, simulando le funzionalità di un sensore. Si possono utilizzare delle impostazioni di default, mentre altre possono essere create ad hoc manualmente tramite i plugin.
- *collision*: caratterizza l'area di collisione di un certo link con gli altri link del modello, permettendo di specificarne il comportamento durante il contatto (come la durezza della superficie o l'attrito) e la geometria.
- *visual*: specifica la geometria visuale del link utilizzata nel rendering grafico. E' a sua volta composta da *material* e *geometry*.
- *plugin*: applicazioni personalizzate che possono essere associate ad uno tra i seguenti elementi: world, model, link o joint. Generalmente possono essere sviluppate in linguaggio Python o C++ utilizzando delle apposite librerie che permettono di interagire con la simulazione, acquisendo dati, modificando parametri o generando delle forze. [17]

Nel seguente progetto con la nomenclatura "Gazebo" si vuole indicare l'attuale Gazebo Classic nella sua versione numero 11[2].

2 Nel tempo, partendo dal codice sorgente di Gazebo, è stato sviluppato un nuovo ambiente simulativo simile, con lo scopo di renderlo più estendibile ed avanzato. Questo, nominato inizialmente Ignition, venne sviluppato parallelamente a Gazebo per anni. Tuttavia, nell'aprile del 2022 venne annunciata la chiusura del progetto Gazebo, arrivato oramai alla sua ultima versione numero 11, scegliendo di proseguire lo sviluppo solamente del progetto Ignition. Inoltre, per problemi legali dovuti all'uso del nome, nella stessa data il progetto Ignition venne rinominato Gazebo, prendendo effettivamente il posto del suo ormai predecessore, mentre all'applicazione originale venne dato il nome di Gazebo Classic per distinguere le due. [15]

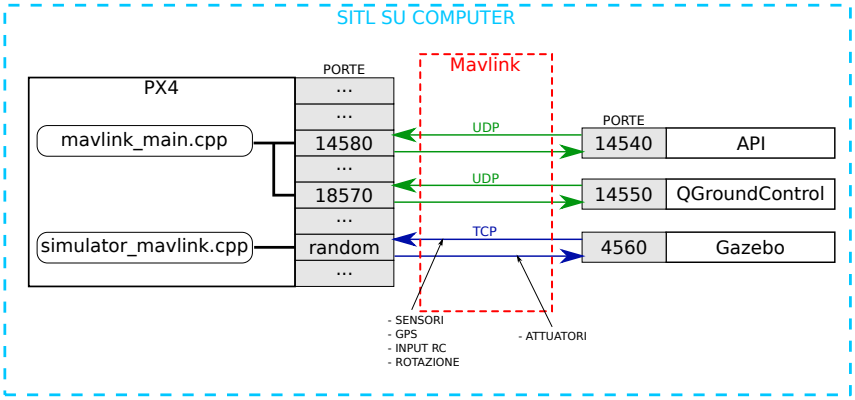

Figura 33: Schematizzazione della comunicazione all'interno di un sistema SITL simulato con Gazebo

4.2.2 *Simulazione con PX4 Autopilot*

PX4 Autopilot, oltre ad essere un firmware, permette anche la simulazione dei veicoli utilizzando un computer e offrendo una serie di ambienti e modelli preimpostati. Tra i vari simulatori compatibili è presente Gazebo, i cui modelli predefiniti e la relativa struttura software si possono trovare nella cartella *Tools/sitl-gazebo*. PX4 permette di avviare due tipologie di simulazione:

- SITL (*Software In The Loop*): la simulazione viene eseguita esclusivamente nel computer, compreso il firmware di PX4. Quest'ultimo comunica con Gazebo server tramite degli appositi plugin, riuscendo a ricevere quindi i dati dalla simulazione come se provenissero dai sensori reali (Figura 33).
- HITL (*Hardware In The Loop*): il firmware di PX4 viene eseguito all'interno di un controllore di volo reale (come Pixhawk 4), il quale scambia informazioni con la simulazione tramite una connessione USB o equivalenti. Tutti i dati relativi alla sensoristica, che di solito è a bordo del veicolo, vengono ricavati dalla simulazione sempre tramite plugin. Il controllore effettua i relativi calcoli come se fosse realmente in volo, ma restituendo i dati direttamente al simulatore.

I modelli Gazebo dei veicoli, per comunicare al firmware i dati derivanti dal motore fisico, sfruttano una serie di plugin i cui relativi codici sorgente risiedono all'interno della cartella *src*. I più utilizzati sono:

- *gazebo_barometer_plugin.cpp*: a partire dal dato di posa[3] del veicolo ne trasforma l'altitudine in un valore equivalente di pres-

3 Tradotto dall'inglese *pose*, con posa si intendono le informazioni relative alla posizione di un certo oggetto espresso nelle coordinate cartesiane rispetto ad sistema di riferimento globale, e la sua rotazione espressa come quaternione.

sione barometrica, simulando anche eventuali dipendenze dalla temperatura;

- *gazebo_gps_plugin.cpp*: converte la posa del veicolo in dati equivalenti ad un GPS;
- *gazebo_imu_plugin.cpp*: ricava tutti i dati di accelerazione lungo i tre assi cartesiani;
- *gazebo_magnetometer_plugin.cpp*: ricrea i dati di un magnetometro equivalente, fornendo le relative misure del campo magnetico;
- *gazebo_motor_model.cpp*: a partire dai target impostati per ciascun attuatore, simula il comportamento fisico delle eliche introducendo le rispettive forze al modello. Si occupa quindi principalmente di fornire la spinta dei rotori al veicolo e di visualizzarne la rotazione.

Ciascuno di questi plugin simula al proprio interno anche gli eventuali rumori aleatori di misura. Tutti i dati ricavati vengono trasmessi tramite degli appositi topic ad un altro plugin nominato *gazebo_mavlink_interface.cpp* il quale si occupa di raccoglierli ed interfacciarsi con il firmware (che sia esso virtualizzato o a bordo del controllore reale) tramite il plugin *mavlink_interface.cpp*. La comunicazione avviene tramite dei prestabiliti messaggi e utilizzando una porta TCP, solitamente la numero 4560.

Proprio grazie a questa struttura si è anche in grado di controllare il veicolo tramite QGroundControl, o qualsiasi altro software equivalente, utilizzando un'ulteriore porta UDP (solitamente la numero 14550) e, nel caso di simulazione di tipo HILT, il controllore può ricevere i comandi direttamente dal radiocomando. Per avviare una simulazione SITL nell'ambiente Gazebo si utilizza il comando da terminale:

```
$ make px4_sitl gazebo_<nome_del_veicolo>
```

il quale si occupa di compilare il firmware e tutti i plugin necessari al modello richiesto.

La caratterizzazione di ciascun veicolo è archiviata all'interno della cartella *models*, al cui interno troviamo tutti i modelli già disponibili di default come: Iris (un quadrotor simile al QAV250) e Typhoon h480 (un esarotore di dimensioni considerevoli e provvisto di un meccanismo di movimentazione del sistema di atterraggio), dai quali si è partiti per lo sviluppo del modello del QAV250. Ciascun veicolo è descritto tramite un file in formato SDF, come già presentato in 4.2.1, e da uno o più file di tipo mesh. Questi ultimi rappresentano la geometria 3D da utilizzare all'interno del rendering e possono essere prodotti a partire dal modello CAD del veicolo. La descrizione

del world invece, e di conseguenza anche della configurazione del motore fisico, si può trovare nella cartella *worlds*. [10]

4.2.3 *Creazione modello QAV250*

Per lo sviluppo del modello del QAV250 si è inizialmente partiti dall'Iris già presente all'interno di PX4 Autopilot, andando a modificare le geometrie mesh visualizzate e rimpiazzandole con quelle esportate dal disegno 3D in formato STL e DAE. In una prima fase, allo scopo di rendere il più accurato possibile il modello, si è scelto di considerare ogni singolo pezzo del drone composto dal medesimo materiale o considerabile come un unico pezzo, come un elemento della simulazione a sè stante. Questo permette di assegnare una massa, un momento d'inerzia e un centro di massa specifico per ciascun componente, come già presentato in 4.1.2. Considerando come link genitore la piastra in fibra di carbonio presente nella parte inferiore del drone, tutti gli altri pezzi sono stati fissati ad esso tramite un link di tipo fisso. Tuttavia, per questioni analitiche e allo scopo di riuscire a calcolare delle forze di legame finite all'interno di una divisione temporale discreta, il motore fisico considera ciascun joint, seppur fisso, come un collegamento di tipo molla-smorzatore. La composizione di un numero così elevato di elementi comporta però un gran numero di interazioni a catena che, in alcuni casi, portano a dei fenomeni vibratori o addirittura di instabilità, con la produzione di forze impulsive molto elevate. Seppur ogni interazione molla-smorzatore possa essere configurata dettagliatamente, la scelta di un buon equilibrio che non portasse a nessun tipo di disturbo e ad una buona affidabilità del modello si è rilevata molto complicata. Un joint troppo rigido porta a delle forze di interazione troppo elevate che, in una simulazione a tempo discreto, si presentano come degli impulsi che possono causare scatti e innescare altri fenomeni di instabilità. Un joint troppo poco rigido, seppur più stabile, comporta un'irrealistica compenetrazione tra i vari elementi.

Un miglioramento delle dinamiche in questa situazione specifica si può ottenere solamente irrigidendo ciascun joint e aumentando la frequenza di calcolo del simulatore, sfruttando il motore fisico Bullet, più leggero di ODE. Tuttavia, visti gli scarsi risultati, si è optato per l'utilizzo di un'unica geometria rappresentante il corpo del drone, ad esclusione delle zampe e delle eliche. Questi ultimi elementi, invece, sono stati fissati con dei joint di tipo *revolute* al telaio. Si impiegano i calcoli presentati in 4.1.2, per attribuire al pezzo centrale una massa e un'inerzia equivalente che possa rappresentare l'intero insieme formato da pezzi di vario materiale.

Per quanto riguarda la collisione del drone (Figura 34), si è scelto di implementarla relativamente ai rotori, in quanto sono gli elementi più esposti e che più facilmente impattano con altri oggetti, e al pat-

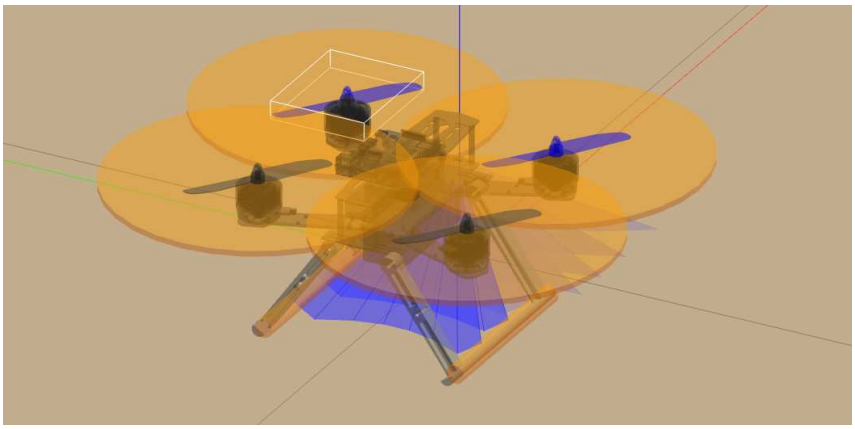

Figura 34: Screenshot di una simulazione in Gazebo nella quale sono evidenziati in arancione i volumi relativi alla collision

tino ci ciascuna zampa i, quali dovranno stare a contatto con il suolo. Nel primo caso (rotori) si utilizza come geometria un disco piatto e di diametro pari al diametro massimo di ingombro delle eliche, mentre per i pattini si preferisce usare dei cilindri. Nonostante Gazebo permetta l'importazione di geometrie per la collision a partire da file mesh, si preferisce basarsi su geometrie elementari per semplificare il carico computazionale e per evitare fenomeni indesiderati a causa della discretizzazione della mesh. La presenza del TOF ad 8x8 zone si è modellizzata utilizzando una tipologia di sensori già presente all'interno di Gazebo, nominata *ray*, configurando la presenza di 8 raggi all'interno del tag <horizontal> e 8 raggi in <vertical>. Per entrambe le direzioni si sono impostate delle ampiezze angolari di ± 0.7854 *rad* corrispondenti ai $\pm 45°$ definiti come range minimo nel datasheet.

4.2.4 *Sviluppo dei plugin Gazebo*

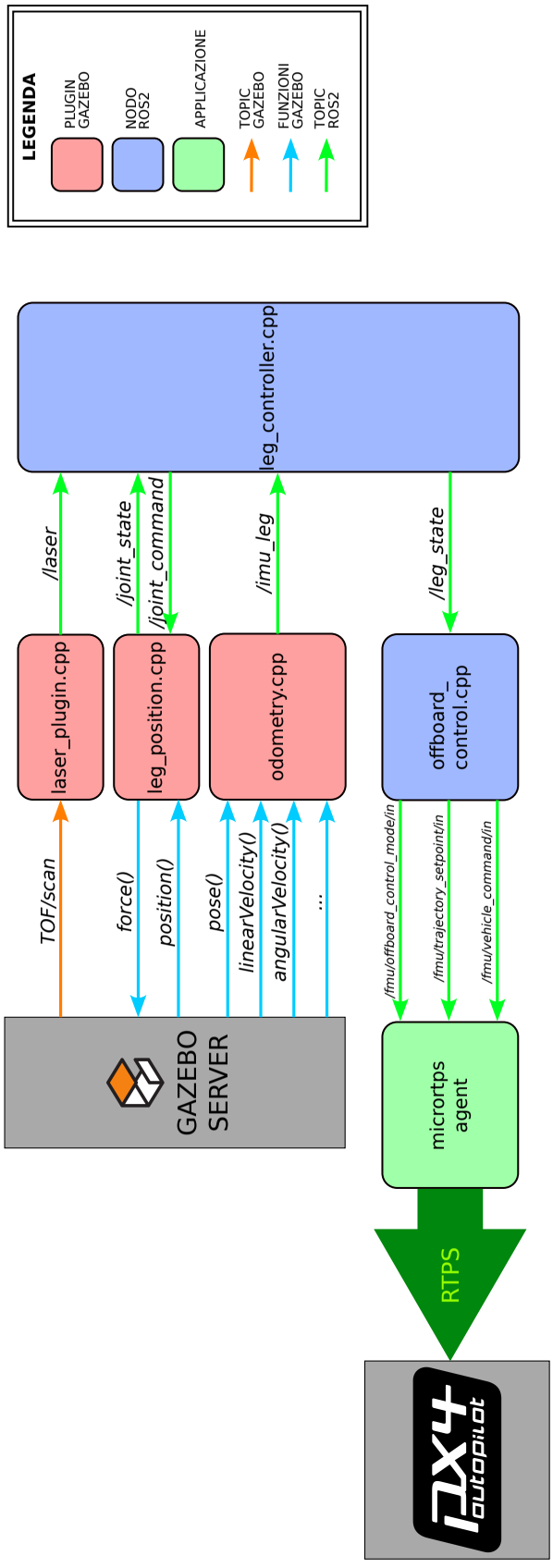

Figura 35: Schematizzazione dell'insieme di plugin, nodi ROS2 e altri software implementati e sviluppati durante il progetto

Per poter interagire con l'ambiente Gazebo e riuscire a ricavare i dati necessari al controllo del sistema che si sta sviluppando, è necessario aggiungere al modello alcuni plugin creati appositamente. Questi si occupano principalmente di interfacciarsi con l'ecosistema di controllo delle zampe e di navigazione, tramite comunicazione ROS 2, sottoscrivendosi e pubblicando i dati necessari.
Lo sviluppo della simulazione parte dall'ipotesi di poter verificare il corretto funzionamento di ciò che si sta sviluppando prima di andare ad implementarlo all'interno del velivolo reale, e per questo è necessario mantenere una certa struttura che permetta di utilizzare lo stesso codice in entrambi i casi, aspettandosi un comportamento perlopiù similare. Per questo, da una parte si utilizzano dei plugin che vanno a simulare in Gazebo l'azione degli attuatori e le misure dei vari sensori, impiegando delle tipologie di topic che possano essere riportati anche nel caso reale. Dall'altra, si costruisce una struttura unica sia per il caso simulato che per il caso reale, in grado di interfacciarsi con questi dati.

4.2.4.1 *Controllo del posizionamento delle zampe: leg_position.cpp*

Questo primo plugin, nominato *leg_position.cpp*, si occupa di simulare all'interno di Gazebo l'azione del servomotore che agisce sulle zampe. Esso si sottoscrive al topic ROS2 */joint_command*, il quale comunica la posizione target verso la quale il servomotore deve ruotare, in radianti, e compresa tra $-pi/2$ e $pi/2$. Questo è pubblicato dal nodo controllore dell'intero sistema che verrà approfondito più avanti. La rotazione del servomotore verso tale target è modellizzato come un movimento a velocità costante, andando quindi ad aggiornare internamente, ciclo per ciclo, la posizione che va ad assumere, come segue:

```
if(target_ang_ > interm_ang_){ //angolo target >
    attuale, il servo deve ruotare in senso antiorario
double step_target_ang_ = interm_ang_ + SERVO_VEL*
    update_dt; //nuovo posizione del servo

if(step_target_ang_ > target_ang_){ //se la nuova
    posizione supera il target, limitalo
interm_ang_ = target_ang_;
}else{
interm_ang_ = step_target_ang_;
}

}else{  //il servo deve ruotare in senso orario

double step_target_ang_ = interm_ang_ - SERVO_VEL*
    update_dt;

if(step_target_ang_ < target_ang_){
interm_ang_ = target_ang_;
```

```
}else{
interm_ang_ = step_target_ang_;
}

}
```

dove *SERVO_VEL* è la velocità lineare del servo espressa in rad/s, la quale va moltiplicata al tempo ciclo per ricavare l'incremento dell'angolo. Utilizzando i dati calcolati come illustrato in 3.3.2, si ricava, a partire dalla rotazione del servomotore, la posizione equivalente che ciascuna delle zampe deve assumere. I valori di $A1$, $B1$, $C1$, $A2$, $B2$ e $C2$ sono delle costanti che rappresentano i coefficienti determinati tramite l'interpolazione.

```
double left_leg_pos_ = A1*pow(interm_ang_,3) + B1*pow
    (interm_ang_,2) + C1*interm_ang_ + D1;
double right_leg_pos_= A2*pow(interm_ang_,3) + B2*pow
    (interm_ang_,2) + C2*interm_ang_ + D2;

//correzione dell'offset dovuto alla posizioni
    iniziali delle zampe
target_left_leg_ = -(left_leg_pos_ - INITIAL1);
target_right_leg_ = -(INITIAL2 - right_leg_pos_);
}
```

Siccome all'interno di Gazebo non è possibile impostare direttamente una posizione per un link specifico, il movimento deve avvenire tramite la creazione di una coppia su ciascun giunto della zampa. Si ricorre così ad un sistema composto da due PID, uno per zampa, avente in ingresso l'errore di posizione, ovvero la differenza tra il target appena trovato e la vera posa di ciascuna zampa. L'output di ciascun controllore viene imposto come coppia in ciascun giunto sotto forma di vettore con direzione parallela all'asse di rotazione. Il PID viene creato utilizzando la classe *gazebo::common::PID* presente nelle librerie stesse di gazebo che, una volta inizializzato, viene aggiornato ad ogni ciclo in questa maniera:

```
error1 = -target_left_leg_ + left_joint_ -> Position
    (0); //errore di posizione
double force1 = pid1_.Update(error1, update_dt); //
    aggiornamento PID
left_joint_ -> SetForce(0, force1); //Creazione
    coppia sul joint
}
```

Infine, esclusivamente per questioni di debug, questo plugin si occupa anche della pubblicazione di un topic che comunica l'effettiva posizione di ciascuna zampa.

4.2.4.2 *Trasmissione dell'odometria: odometry.cpp*

Per ottenere le informazioni riguardanti l'odometria del veicolo, utili ai sistemi di controllo delle zampe a valle, si sviluppa anche questo semplice nodo, nominato *odometry.cpp*, che si occupa inizialmente di acquisire da Gazebo la posizione, la rotazione, le velocità (sia lineari che angolari) e le accelerazioni del link principale (corrispondente alla struttura centrale del drone e comprendente anche il controllore di volo). Ciò è possibile tramite le apposite funzioni messe a disposizione dalla libreria di Gazebo, le quali vanno ad acquisire le informazioni direttamente dal motore di calcolo. Questi dati vengono poi trasmessi tramite un apposito topic ROS2, il cui messaggio viene definito ad hoc nel file *Imu.msg* e, schematicamente, composto come segue:

- *header*: genericamente presente nella quasi totalità dei messaggi ROS2 e definita all'interno dei messaggi standard di ROS2 (*std_msgs*). Presenta un ID univoco e sequenziale per ciascun messaggio e trasmette le informazioni riguardanti la tempistica e il frame a cui è associato.

- *pos*: posizione del link, definita a partire da un messaggio di tipo *geometry_msgs*. Contiene il vettore cartesiano della posizione del centro geometrico del drone rispetto al sistema di riferimento assoluto e il quaternione dell'orientazione.

- *vel*: vettore a 6 dimensioni riportante la velocità lineare e angolare di movimento del link.

- *acc*: rappresenta l'accelerazione lineare e angolare, similmente al caso precedente.

A differenze dell'odometria già misurata dall'apposito plugin di PX4, non viene simulato alcun tipo di rumore di misura. Nonostante questa accortezza, si può osservare che i dati ottenuti sono tutt'altro che ideali e privi di rumore, caratteristica dovuta all'interazione di vari elementi effettuata con calcoli a tempo discreto. Questa scelta è dettata dal fatto che nel drone reale la posizione è ricavata tramite una fusione di informazioni provenienti da vari sensori, le quali vengono opportunamente filtrate, portando ad avere dei dati sostanzialmente stabili e non particolarmente rumorosi. In questa prima fase di sperimentazione non si vuole quindi creare dei modelli aleatori non realistici e ci si limita a considerare solo i rumori già presenti.

4.2.4.3 *Gestione dati del TOF: laser_plugin.cpp*

Avendo modellizzato il TOF come un sensore all'interno del file SDF, i relativi dati vengono pubblicati da Gazebo stesso tramite un apposito topic nominato *~/qav250/base_link/TOF/scan*, non compatibile direttamente con ROS2. Il plugin *laser_plugin.cpp*, quindi, si occupa

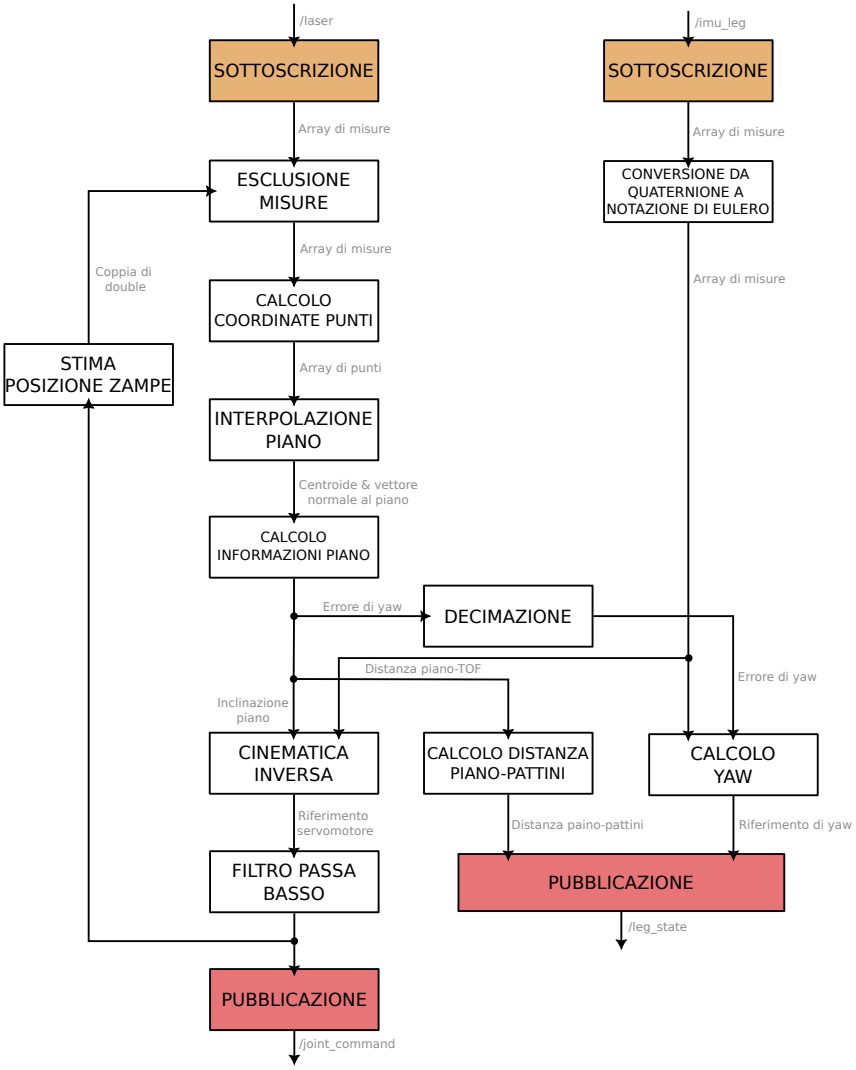

Figura 36: Schema semplificato del funzionamento del nodo leg_controller.cpp ad esclusione dell'elaborazione dei dati di debug

di sottoscriversi a questo topic e ripubblicarlo come */laser* su ROS2. Come per l'odometria, al fine di riuscire a trasmettere all'interno del topic tutte e sole le informazioni necessarie, è stato creato un apposito messaggio. Esso è composto, oltre al classico header, dai dati riguardanti il range di copertura del sensore e da un vettore di 64 elementi rappresentante la misurazione di profondità di ogni singolo raggio.

4.2.5 *Sviluppo nodo ROS 2 di controllo: leg_controller.cpp*

Una volta costruita la struttura di plugin che si interfaccia con Gazebo, si procede all'implementazione di un nodo ROS 2 in grado di controllare l'intero di atterraggio (composto da zampe e servomotore) a partire dai dati raccolti (Figura 36). Questo nodo è indipendente

dalla simulazione e può essere eseguito su qualsiasi elaboratore connesso alla rete locale, anche non montato a bordo del QAV250. Esso è definito nel file *leg_controller.cpp* e nello specifico si occupa di:

- sottoscriversi ai dati provenienti dal sensore TOF, escludendo i dati che risultano poco significativi provenienti da raggi oscurati dalle zampe o fuori dalla portata del sensore;
- eseguire l'interpolazione tridimensionale del piano di appoggio a partire dai dati del TOF ritenuti affidabili;
- ricavare la pendenza stimata del piano sottostante e l'orientazione rispetto al drone tramite calcoli trigonometrici;
- tramite la cinematica inversa, di cui 3.3.3, ricondursi alla posizione che le zampe dovrebbero assumere per poter atterrare sul piano e il rispettivo target per il servomotore;
- inviare il comando di posizionamento tramite il topic */joint_command*;
- stimare la distanza tra i punti di contatto delle zampe e la superficie di atterraggio;
- calcolare il target di yaw che il drone deve assumere per allinearsi al piano e inviarlo al nodo di controllo *offboard*.

Riguardo a quest'ultimo punto, va specificato che l'obbiettivo di tale nodo, oltre ad una gestione delle misure del TOF, è di controllare il movimento delle zampe a dovere e di allineare il drone con l'orientazione del piano. Infatti, come da ipotesi del progetto, il sistema attuato permette la variazione dell'inclinazione della struttura di atterraggio soltanto lungo una direzione e quindi riuscendo ad atterrare correttamente solo nel caso il piano sia inclinato nella direzione ortogonale a quella longitudinale del drone. Il presente nodo deve quindi anche occuparsi di allinearsi all'orientazione del piano tramite l'azione sullo yaw in fase di atterraggio.

4.2.5.1 *Esclusione misurazioni non affidabili*

Il posizionamento del TOF sulla superficie inferiore della struttura del drone crea delle problematiche dovute all'oscuramento di alcuni raggi laser da parte delle zampe in maniera variabile in funzione della posizione delle stesse. Per questo motivo, per non avere degli scompensi nella corretta interpolazione del piano, è necessario escludere tutte quelle misurazioni di profondità che si ritengono distorte da questo fenomeno. Si ipotizza inizialmente che il pattino, essendo parallelo alla direzione longitudinale del velivolo, vada ad oscurare tutti i raggi appartenenti ad una stessa colonna di laser, la quale è variabile a seconda della posizione della zampa. A partire da queste ipotesi è possibile studiare il fenomeno in maniera bidimensionale,

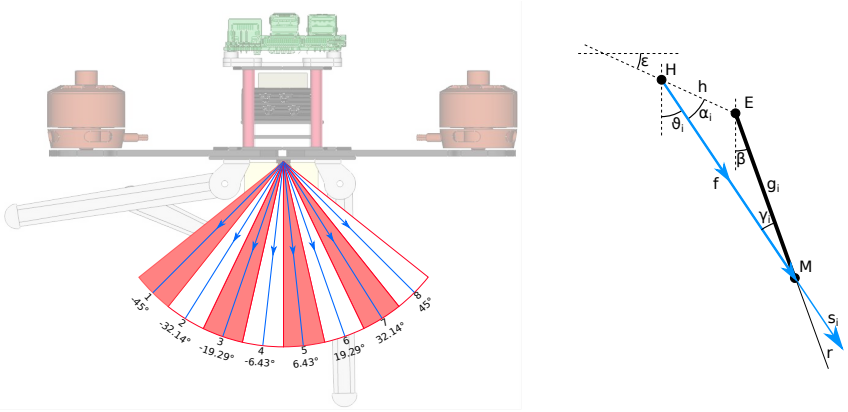

Figura 37: Schematizzazione della suddivisione delle aree di ciascun raggio (a sinistra) e semplificazione del caso in un modello geometrico bidimensionale (a destra)

assegnando a ciascuna colonna un'area di azione definita come in figura 37 (a sinistra). Una colonna si considera oscurata se il pattino passa all'interno della relativa zona.
Per trattare analiticamente il problema, si considera dapprima la zampa di destra e, infine, per simmetria si riportano gli stessi risultati anche per la zampa di sinistra. Si consideri una retta r (Figura 37 a destra), passante per il centro di rotazione della zampa E, parallela alla zampa stessa e β la rotazione della zampa destra (conosciuta tramite stima). Si consideri, inoltre, la retta s_i come la retta rappresentante il limite superiore della i-esima zona di oscuramento e si appone lo stesso indice i a tutte le grandezze relative al medesimo limite. Per costruzione vale:

$$\gamma_i = \theta_i - \beta \tag{37}$$

e

$$\alpha_i = 90^\circ - \theta_i - \epsilon \tag{38}$$

Sia H il punto da cui partono i raggi prodotti dal sensore TOF. Conoscendo le coordinate in x e y dei punti E ed H, noti dal disegno tecnico, ci si ricava i seguenti valori:

$$\epsilon = \text{atan2}(y_H - y_E, x_H - x_E) \tag{39}$$

$$h = \sqrt{(y_H - y_E)^2 + (x_H - x_E)^2} \tag{40}$$

dove ϵ è l'angolo compreso tra il piano principale del drone e la proiezione del segmento EH, mentre h è la lunghezza di quest'ultimo.

Tramite il teorema del seno si può ricavare il valore di g_i, ovvero la lunghezza della proiezione del raggio laser sulla retta r:

$$\frac{h}{\sin \gamma_i} = \frac{g_i}{\sin \alpha_i} \tag{41}$$

che quindi sarà:

$$g_i = h \frac{\sin \alpha_i}{\sin \gamma_i} \tag{42}$$

Di conseguenza, la condizione di oscuramento della i-esima zona avviene se

$$0 < g_i < l < g_{i-1} \tag{43}$$

dove l rappresenta la lunghezza della zampa. Intuitivamente, la diseguaglianza 43 indica che l'i-esimo raggio viene oscurato se il pattino, posto all'estremità della zampa, entra all'interno della relativa zona. Questo può essere tradotto geometricamente considerando la proiezione sulla zampa del limite superiore dell'i-esima zona. Se questa è di lunghezza inferiore rispetto alla zampa, il limite superiore è stato valicato. Il controllo viene quindi effettuato anche sul limite inferiore, il quale coincide con il limite superiore della zona successiva.

Implementando questo calcolo all'interno di Octave si può ricavare un vettore rappresentante l'orientazione che la zampa assume nella condizione limite delle zone di oscuramento. Gli stessi valori possono essere impiegati così come sono anche per la zampa sinistra, con l'accortezza di aggiustare l'indice in maniera speculare. I valori ottenuti analiticamente sono riportati all'interno del nodo controllore nel quale si è creata un'apposita funzione nominata *ExclRay* la quale fornisce l'indice della colonna delle misure da escludere, fornendone in ingresso la posizione della zampa. Oltre a questo, il numero di misurazioni da considerare va ulteriormente decimato, escludendo i casi in cui il piano è ad una distanza troppo elevata e fuori dalla portata del sensore. Per questo motivo viene implementato un ulteriore controllo che esclude tutti i raggi che riportano una distanza superiore ai $3,5\ m$, valore sufficientemente elevato per i nostri scopi e ben all'interno della portata massima del TOF in questione.

4.2.5.2 *Interpolazione del piano di atterraggio*

Una volta escluse le misurazioni non attendibili secondo quanto detto nel paragrafo precedente, si ottiene una serie di dati riguardanti le distanze di alcuni punti del piano sottostante il veicolo rispetto alla posizione del TOF. Per ciascuno di questi raggi se ne conosce la relativa

numerazione di riga i e colonna j e ne è quindi facilmente calcolabile l'orientazione, in entrambe le direzioni, rispetto alla verticale:

```
alfa_ = LEFTLIM + STEPANG *i;
beta_ = LEFTLIM + STEPANG *j;
```

dove LEFTLIM è l'angolazione del primo raggio della colonna e della riga, mentre STEPANG è l'angolo compreso tra due raggi adiacenti nella stessa riga o nella stessa colonna nell'ipotesi che siano equamente distribuiti all'interno di tutto il range. Salvando la misura di ciascun raggio nella variabile *laser_*, si possono ricavare le coordinate del relativo punto di collisione del laser con il piano rispetto al sistema di riferimento con centro nel TOF:

```
temp_.x() = laser_*sin(alfa_);
temp_.y() = laser_*sin(beta_);
temp_.z() = -laser_*cos(alfa_)*cos(beta_);
points_.push_back(temp_);
```

Ciascuno di questi punti, espressi nelle coordinate cartesiane, appartiene al piano di atterraggio e viene salvato in un'apposito array contenente tutti i valori ottenuti ad ogni ciclo di misurazione del TOF.
Dal momento che la misura, anche in ambiente simulato, è affetta da rumore ed errori, ed avendo a disposizione un certo numero n di misurazioni, con $2 < n \leq 64$, è necessario creare un'interpolazione mediata di tutti i dati a disposizione. Più elevata sarà la quantità di dati raccolti e più la stima sarà accurata.

Nel caso ideale di misurazioni prive di rumore, per l'interpolazione di un piano, sono necessarie e sufficienti le coordinate di 3 punti, mentre qualsiasi ulteriore valore risulterebbe ridondante. Tuttavia, partendo da misurazioni rumorose e ad incertezza non nulla, si utilizza una stima ai minimi quadrati in grado di mediare tutti i dati a disposizione e ricavare un piano che minimizzi le distanze dei punti dal piano stesso. In linea teorica si ipotizza che il piano possa essere completamente caratterizzato mediante un punto (x_0, y_0, z_0) ad esso appartenente ed un vettore (a, b, c) ad esso normale. La distanza di un generico punto $P_i = (x_i, y_i, z_i)$ dal piano si può esprimere come:

$$d_i = a(x_i - x_0) + b(y_i - y_0) + c(z_i - z_0) \tag{44}$$

Sia questo punto uno degli N (con $N = 64$ in questo caso) valori ricavati precedentemente a partire dalle misure del laser, ci si pone di minimizzare la seguente funzione:

$$E = \sum_{n=1}^{N} d_i^2 \tag{45}$$

la quale è funzione dei parametri del piano. Senza entrare nei dettagli, questo tipo di operazione di ottimizzazione può essere svolta

tramite la decomposizione ai valori singolari, la quale ben si presta ad un'implementazione all'interno di un algoritmo. In particolare i passaggi da effettuare per questo tipo di operazione sono:

- calcolo del centroide $(\bar{x}, \bar{y}, \bar{z})$ dai dati a disposizione, svolto come media di ciascuna coordinata:

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_i \tag{46}$$

$$\bar{y} = \frac{1}{N} \sum_{n=1}^{N} y_i \tag{47}$$

$$\bar{z} = \frac{1}{N} \sum_{n=1}^{N} z_i \tag{48}$$

- sottrazione a ciascun punto del valore del centroide al fine di agevolare la decomposizione ai valori singolari;
- creazione della matrice A, di dimensione $Nx3$, composta dai valori trovati al punto precedente disposti in riga;
- decomposizione ai valori singolari della matrice A.

Il vettore (a, b, c) corrisponde ai 3 più piccoli valori singolari tra quelli ricavati, mentre il punto (x_0, y_0, z_0), caratterizzante il piano, è il centroide stesso. [2]

4.2.5.3 *Calcolo informazioni dal piano*

Dall'interpolazione effettuata a partire dalle misure, riusciamo a stimare, oltre alla distanza del piano dal TOF, il vettore normale alla superficie, il quale ci fornisce implicitamente le informazioni riguardanti l'orientazione e l'inclinazione del piano rispetto al sistema di riferimento del drone, rispettivamente σ e τ. Conoscendo le coordinate cartesiane del vettore normale alla superficie (a, b, c), i due valori possono essere ottenuti come:

$$\sigma = \operatorname{atan2}(a, b) \tag{49}$$

$$\tau = -\operatorname{atan2}(b, -c) \tag{50}$$

E' quindi possibile ricavare il riferimento di yaw da inviare al controllore di volo, il quale, tuttavia, va ad indicare l'orientazione del drone rispetto ad un sistema di riferimento assoluto, tipicamente solidale al campo magnetico terrestre. Si può quindi definire σ come

un errore di yaw rispetto all'orientazione voluta e allineata al piano inclinato. Si utilizza quindi la misura dello yaw ottenuta tramite il nodo di odometria, presentato precedentemente (sottoparagrafo 4.2.4.2), andando a sommare l'errore, ovvero σ, e correggendo eventuali segni e offset dovuti a rotazioni dei sistemi di riferimento:

```
yaw_ = - orientation_.z + yaw_incr_ + 1.5707;
```

4.2.5.4 *Implementazione cinematica inversa*

Tramite le considerazioni riportate in 3.3.3 sappiamo che il legame tra inclinazione del piano e relativo posizionamento del servomotore è approssimabile come una funzione lineare nell'ipotesi che il drone sia in posizione orizzontale. Questa condizione è verificata soltanto se il drone è in quiete e in assenza di perturbazioni (come ad esempio folate di vento). Per ricondursi ad un caso più generico, è necessario considerare piuttosto l'angolo che viene a formarsi all'intersezione tra il prolungamento del piano di atterraggio e quello del drone. Con questa tecnica, l'orientazione delle zampe verrà comandata in modo da compensare anche eventuali inclinazioni del drone. All'interno del nodo si va ancora una volta ad utilizzare i dati trasmessi dal nodo odometria, il quale ci fornisce una misura del roll del drone, per poi andare a calcolare la cinematica inversa. Il valore che si ottiene è equivalente all'angolo target del servomotore che, però, risulta affetto da rumori propagati a partire dalle misure a monte ed è quindi necessario filtrarlo adeguatamente allo scopo di evitare dinamiche oscillatorie, tipicamente vibrazioni, sul movimento dell'azionamento. Si utilizza quindi un semplice filtro passa basso del primo ordine implementato a tempo discreto e tarato in maniera qualitativa, il quale risulta più che efficace al suddetto compito. Il valore finale ottenuto a valle di questi passaggi viene pubblicato sul topic */joint_command* e destinato al nodo di controllo del servomotore.

4.2.5.5 *Calcolo distanza tra superficie e zampe*

Allo scopo di rilevare il contatto a terra del drone, comandandone il disarmo, e per ottimizzare la dinamica di atterraggio in modo da rallentare la velocità di discesa quando si è prossimi al contatto, risulta necessario ricavare un valore di distanza dalla superficie. Questo dato è stato calcolato a partire dalla informazione di distanza tra piano inclinato e sensore TOF proveniente dall'interpolazione del piano, che quindi non tiene conto della presenza delle zampe. Si è calcolata quindi una stima della posizione delle zampe, α e β, a partire dal comando al servomotore e sfruttando ancora una volta l'approssimazione della cinematica diretta. Note le coordinate cartesiane del centro di rotazione dei perni delle zampe, rispettivamente nei punti D ed E,

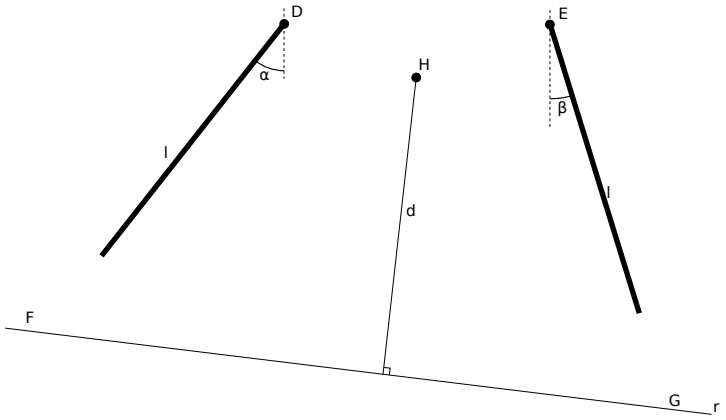

Figura 38: Schematizzazione semplificata della geometria per il calcolo della distanza del piano dal punto di contatto con le zampe

si ricava le coordinate di F e G corrispondenti ai punti di contatto con la superficie, riportandoci al caso bidimensionale (Figura 38):

$$\begin{cases} x_G = x_E + l\sin(\beta) \\ y_G = y_E - l\cos(\beta) \\ x_F = x_D - l\sin(\alpha) \\ y_F = y_D - l\cos(\alpha) \end{cases} \tag{51}$$

Si può quindi ricavare l'equazione della retta passante per questi due punti, il cui coefficiente angolare e intercetta risultano rispettivamente:

$$m = \frac{y_G - y_F}{x_G - x_F} \qquad q = y_F - mx_F \tag{52}$$

Definendo il punto H coincidente con il centro equivalente del sensore TOF, si è in grado di calcolarne la distanza con la retta appena trovata tramite la formula:

$$d(H, r) = \frac{|y_H - (mx_H + q)|}{\sqrt{1 + m^2}} \tag{53}$$

Questo valore corrisponde quindi alla distanza che intercorre tra il sensore TOF e il piano di atterraggio nel caso di pieno contatto. E' quindi sufficiente sottrarre questo dato appena ottenuto, funzione della posizione delle zampe, con la distanza ricavata tramite l'interpolazione per ottenere la distanza tra il punto di contatto delle zampe ed il piano di atterraggio, ipotizzando che questi siano già allineati per azione dell'attuazione.

4.2.6 *Controllo offboard*

Per effettuare dei voli di prova in simulazione, non potendo collegare un radiocomando nella modalità SITL, si implementa un sistema di comunicazione offboard sfruttando l'interfaccia RTPS. In questo modo si è in grado di inviare al controllore eventuali target di posizione che si vuole raggiungere e leggere informazioni dai sensori a bordo.

4.2.6.1 *Attivazione RTPS nella simulazione*

Nella sottosezione 2.3.2 è stato presentato il sistema di comunicazione RTPS usato a bordo dei velivoli ma che può essere impiegato anche nel software simulato. Per attivare la comunicazione di tipo RTPS è necessario apporre la dicitura "_rtps" all'apposito comando di compilazione e avvio della simulazione, come segue:

```
$ make px4_sitl_rtps gazebo_<nome_del_veicolo>
```

In alcune situazioni può accadere che il controllore attivi la modalità *failsafe* quando, tramite RTPS, si va ad attivare la modalità di offboard. Cioè comporta l'impossibilità da parte del drone di ricevere i comandi di posizione da seguire nonostante risulti già armato. Per ovviare a questo problema si può disattivare questa procedura di sicurezza andando a modificare all'interno di PX4 il parametro *COM_RCL_EXCEPT*. [16]

4.2.6.2 *Il nodo di controllo offboard: offboard_control.cpp*

All'interno del progetto, l'impiego della comunicazione RTPS risulta conveniente per inviare al controllore di volo la traiettoria che si vuole intraprendere e per la gestione dello yaw. Come già illustrato nei paragrafi precedenti, per eseguire correttamente l'atterraggio su delle superfici con inclinazione generica, si è scelto di allineare il drone tramite il controllo dello yaw. Il nodo di *leg_controller*, già presentato in precedenza, si occupa anche di pubblicare su un apposito topic lo yaw di riferimento che il drone deve realizzare e la distanza del pattino dalla superficie, andando a decimare i dati per limitarne la banda passante ed evitare la propagazione di dinamiche oscillatorie. Il nodo *offboard_control* in questione si sottoscrive a questo topic e si occupa di comunicare a PX4 la traiettoria. Al fine di testare il funzionamento dell'intero sistema presentato in questa tesi si aggiunge, all'interno del modello del world della simulazione, tre piani di inclinazione pari a circa 20° ma orientati in maniera casuale (Figura 39). Si crea quindi un algoritmo, all'interno del nodo di offboard_control, il quale si occupa di gestire la traiettoria secondo le seguenti fasi:

- attesa di un tempo sufficientemente elevato per permettere il caricamento del firmware, attivazione della modalità di offboard e armo dei motori (Figura 40 immagine 1);

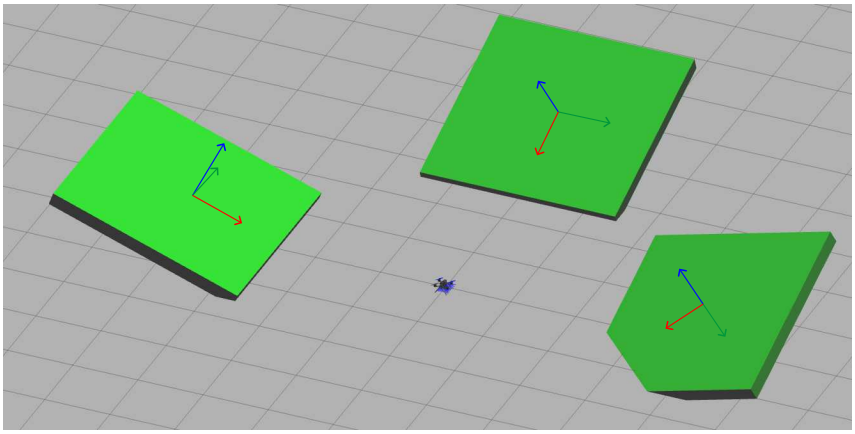

Figura 39: Screenshot della simulazione di Gazebo in cui sono presenti i tre piani di atterraggio

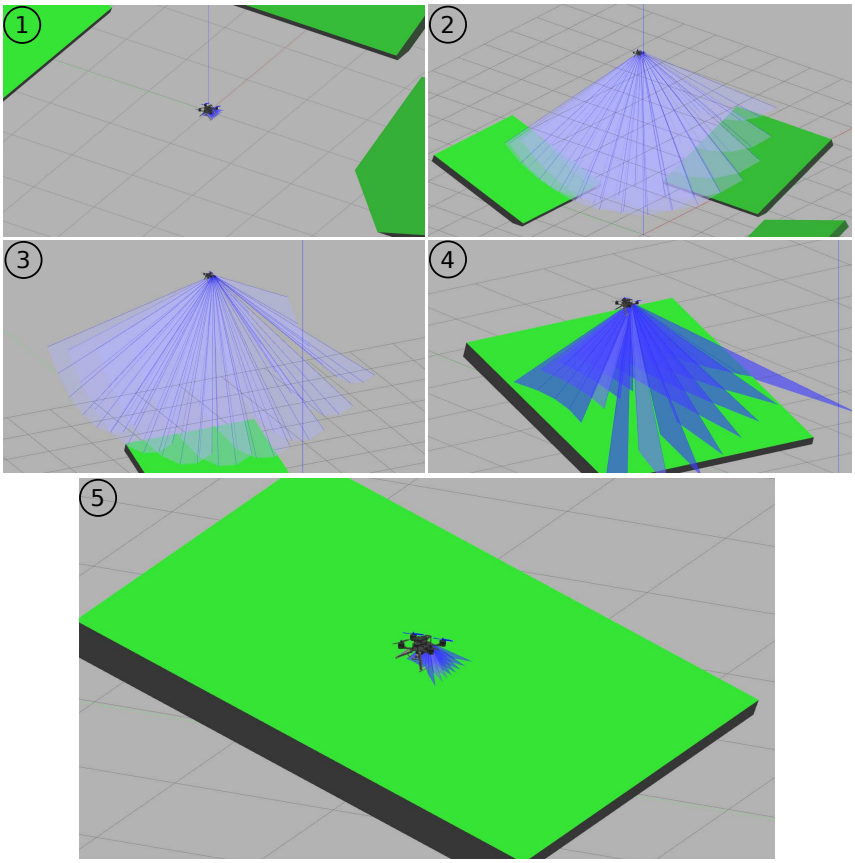

Figura 40: Vari screenshot del rendering di Gazebo durante la simulazione di test con controllo offboard

- decollo e salita verticale fino a quota di 5 *m* (Figura 40 immagine 2);
- spostamento orizzontale fino al posizionamento sopra al primo piano di posizione nota (Figura 40 immagine 3);
- discesa a velocità controllata e variabile a tratti, inizialmente elevata e man mano sempre più lenta all'avvicinarsi alla superficie (Figura 40 immagine 4);
- verifica del contatto con il terreno. Ciò è implementato controllando che la distanza con la superficie rimanga ad un valore prossimo a zero per un tempo pari a 5 *s* (Figura 40 immagine 5);
- decollo e ripetizione dei passaggi precedenti per gli ulteriori due piani;

Durante la navigazione, lo yaw è controllato mediante i dati ricevuti tramite il topic. Nel caso in cui il drone si trovi ad altitudini troppo elevate per poter ottenere una stima del piano sottostante, lo yaw viene mantenuto costante all'ultimo valore ricevuto.

4.2.7 *Aggiunta del meccanismo*

Un'ulteriore passo per la creazione di un modello simulato realistico comprende l'implementazione dell'intero meccanismo all'interno di Gazebo. Si è scelto di effettuarlo in un secondo momento in quanto, come già illustrato precedentemente, l'interazione tra un numero considerevole di link può comportare fenomeni non voluti e quindi si preferisce seguire uno sviluppo passo per passo. Fino a questo punto, il moto alle zampe è fornito tramite una coppia applicata direttamente nel perno di ciascuna zampa, mentre il plugin leg_position.cpp si occupa di ricavare analiticamente la rotazione che le zampe dovrebbero assumere a partire dalla rotazione del servomotore, andando quindi a simulare il vincolo di moto creato dal meccanismo. L'obbiettivo di questo punto è la creazione di questi vincoli fisicamente all'interno del modello Gazebo, andando ad importare le geometrie create tramite disegno 3D e imponendo dei joint di tipo revolute. Il plugin viene modificato in modo da agire con la coppia direttamente sul perno del servomotore, tramite l'utilizzo del solito PID e prendendo come target il valore comunicatogli direttamente tramite il topic /joint_command. E' mantenuta la transizione del servo a profilo a velocità angolare costante. Questa modifica, inoltre, permette di ricavarci dei dati importarti riguardanti le forze in gioco tra i vari joint, i quali possono risultare utili per uno studio dei carichi nei materiali, tramite FEM, portando quindi ad eventuali modifiche delle geometrie al fine di prevenirne la rottura.

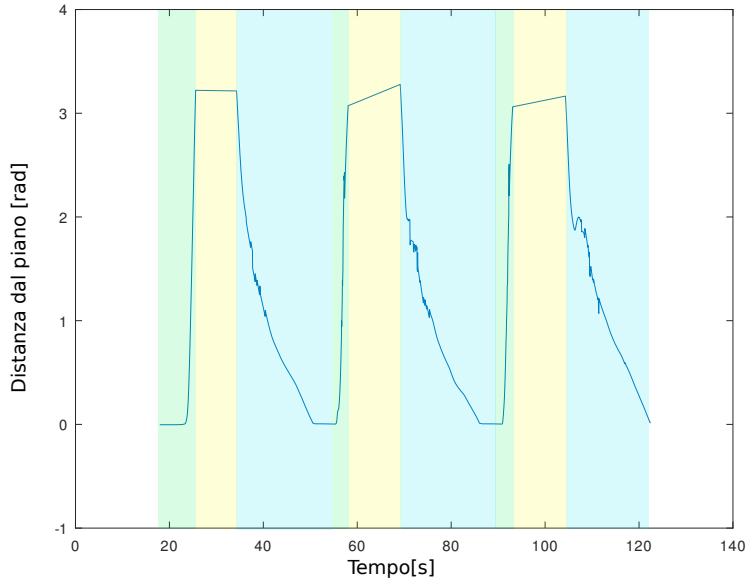

Figura 41: Grafico della distanza dalla superficie stimata tramite misure del TOF

4.3 RISULTATI DELLA SIMUAZIONE

4.3.1 *Dati valutativi della simulazione finale*

A conclusione del lavoro di simulazione si può analizzare se gli obbiettivi posti all'inizio del lavoro di ricerca e prototipazione risultino soddisfatti in maniera adeguata. Il progetto, tuttavia, risulta essere allo stato embrionale e quindi si potrebbe definire prematura la valutazione dei risultati in maniera oggettiva e analitica. Il simulatore però permette di raccogliere informazioni con estrema facilità e senza richiedere l'implementazione di sensoristica avanzata, facilitando quindi lo studio dei fenomeni in gioco. Si procede quindi utilizzando una simulazione di test presentata in 4.2.6.2 e rappresentata in Figura 39, la quale riesce a provare il funzionamento dell'intero sistema in tre casistiche generiche.
Nel grafico in Figura 41 è rappresentata la stima della distanza del piano dal punto di contatto delle zampe, dal quale si possono evidenziare le seguenti caratteristiche:

- Durante il decollo (tratto verde) la misura aumenta velocemente, sintomo dell'estrema reattività del drone il quale riesce a sollevarsi da terra in tempi brevissimi, nell'ordine del secondo. Ciò produce un grafico crescente seghettato dovuto al periodo di campionamento dei dati del laser elevato rispetto alla dinamica del decollo.

- Nella zone gialla l'altitudine effettiva del drone è troppo elevata

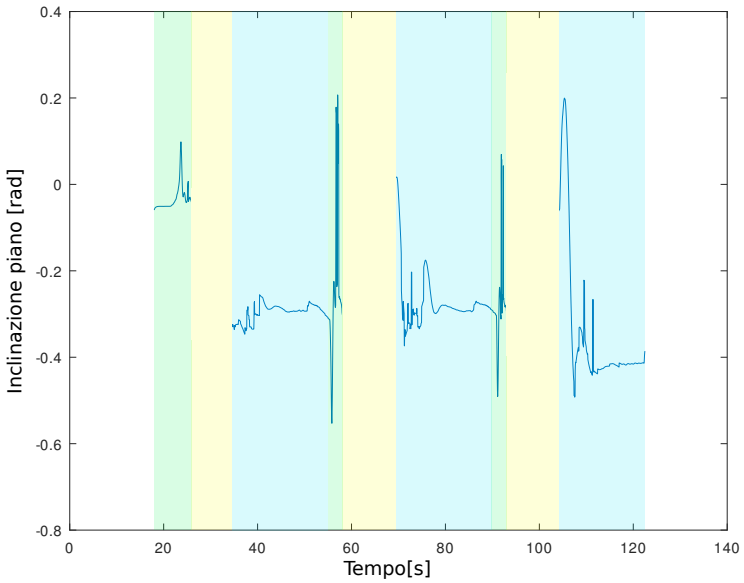

Figura 42: Grafico dell'orientazione del piano stimata tramite misure del TOF

per riuscire ad avere delle misure affidabili dal TOF oltre che risultare inutili ai fini del controllo. Il valore viene quindi limitato ad un valore predefinito e slegato dalle misure.

- La fase di atterraggio (zona blu) è caratterizzata da una discesa brusca nel primo tratto per poi andare a diminuirne la velocità man mano che la distanza dal piano diminuisce.
- Il valore della distanza arriva effettivamente a 0 al momento di contatto del drone con la superficie e rimane sufficientemente stabile per l'intero periodo di tempo.
- Si notano dei rumori in fase di atterraggio, soprattutto nell'ultima occorrenza, nella prima fase di discesa ad alta velocità. Confrontandolo con il grafico delle stime in Figura 42, si nota che il rumore ha afflitto anche la stima dell'orientazione del piano a causa dell'elevata distanza del piano che ha reso più incerta la misura del sensore TOF. Inoltre, avvicinandosi al piano, si verifica un momento in cui solo le misure centrali saranno dentro al range massimo misurabile, mentre gli altri raggi condurranno a misure fuori scala. La diminuzione del numero di dati con i quali eseguire l'interpolazione porta ad una maggior imprecisione del calcolo.

Analizzando il grafico relativo all'orientazione del piano (Figura 42), oltre a quanto già citato, si evidenzia la presenza di picchi elevati, sintomo di forte distorsione, nei momenti di decollo. Tuttavia ciò non risulta influente nel funzionamento, tanto più perchè filtrato a valle

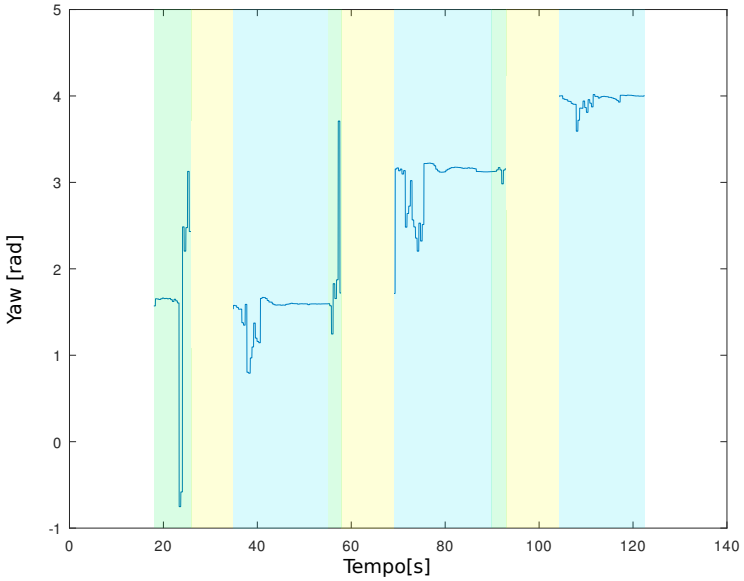

Figura 43: Grafico del target di yaw calcolato a partire dalle misure del TOF

tramite un filtro passa basso prima di arrivare al comando del servomotore. Nei tratti in cui le misure del TOF vengono considerate non valide (zona gialla con altitude superiore ai 3 m), la stima del piano non può essere effettuata per mancanza di dati e quindi si tiene valido il valore ricavato precedentemente a questo evento. Si considerano tuttavia corrette le stime nelle fasi si atterraggio, coerenti con i valori di inclinazioni impostati nel modello dei tre piani.

Si osserva un comportamento simile anche per quanto riguarda lo yaw (Figura 43), con forti distorsioni nelle zone verdi ma valori sufficientemente precisi e congrui all'attesa nelle fasi di atterraggio. Anche qui, il valore di yaw viene imposto costante negli istanti in cui l'altitudine è troppo elevata, in particolare viene mantenuto l'ultimo valore calcolato così da minimizzare le rotazioni del drone. Si riporta infine il grafico rappresentante il target di posizione destinato al servomotore e calcolato dal nodo leg_controller a valle del filtro passa basso, paragonato alla posizione che realmente assume (Figura 44). I due tratti risultano pressochè congruenti nelle situazioni in cui il valore del target non presenta oscillazioni ampie, mentre si risente della velocità di rotazione finita nei casi di bruschi cambiamenti. Tuttavia, analizzando fase per fase, si osserva che questi discostamenti si presentano soltanto in fase di rapido decollo o quando il drone si trova ad elevata distanza dal piano di atterraggio. Ciò, quindi, non influisce sul corretto funzionamento del sistema ed è dovuto alla variazione brusca del pitch e del roll al fine di inseguire il target di posizione imposto dal nodo di offboard.

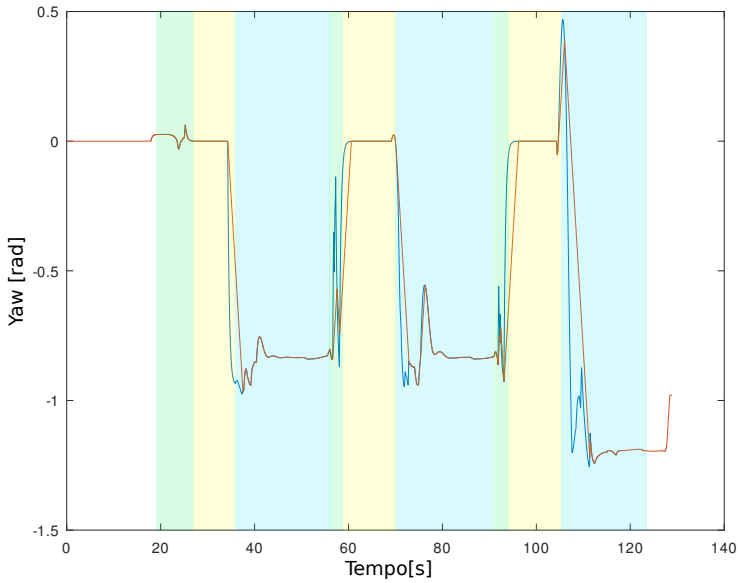

Figura 44: Grafico dell'orientazione reale del perno del servomotore (arancione) e del target inviato dal nodo leg_controller

4.3.2 *Criticità nella preparazione della simulazione*

La preparazione del modello da utilizzare all'interno di Gazebo è risultata molto problematica e dispendiosa di tempo, ponendo svariati problemi che si possono riassumere brevemente come segue:

- L'interfaccia di Gazebo è poco sofisticata e povera di strumenti per il debug. Ciò rende difficile la risoluzione dei problemi dovendo optare per la creazione di appositi plugin di logging dei dati su file, per poi andarli ad elaborare con software di calcolo. Potrebbe risultare utile la presenza di qualche tool per la visualizzazione grafica delle forze di interazione, così da permettere di capire in tempi brevi quali possono essere le cause di eventuali problemi.
- L'editing dei modelli già esistenti tramite l'apposito editor implementato in Gazebo è difficoltosa e spesso va a complicare il file SDF aggiungendo tag non richiesti. E' poco compatibile con i plugin. Questo ha comportata il dover creare manualmente il file di configurazione andando a verificare volta per volta il funzionamento di quanto aggiunto.
- Le simulazioni non sempre funzionano correttamente al primo avvio e presentano frequentemente dei bug, i quali comportano il dover riavviare la simulazione.
- La configurazione dei joint e del motore fisico è difficoltosa in quanto non si riesce a risalire ad una modellizzazione ac-

curata del sistema reale. Infatti, risulta impossibile conoscere in maniera precisa modello massa-molla-smorzatore equivalente dei legami di tipo fisso tra parti, comportando quindi una configurazione per tentativi.

- Proprio a causa della configurazione complicata si creano spesso instabilità o fenomeni non realistici che vanno valutati caso per caso andando a modificare i parametri del modello.
- L'ambiente renderizzato è difficile da navigare nella sua vista tridimensionale.
- La documentazione per la programmazione di plugin alcune volte non è aggiornata e quindi è necessario cercare nel web degli esempi di codice che contengano la funzione desiderata.

5 REALIZZAZIONE SPERIMENTALE

Avendo ottenuto dei buoni risultati a livello simulativo, si procede con una prima fase di costruzione del meccanismo fisico da implementare sul drone al fine di effettuare dei primi test sui pezzi prodotti tramite disegno CAD. In particolare, gli obbiettivi che si vogliono raggiungere con questo tipo di prove sono:

- valutare il comportamento a flessione delle zampe, della struttura e del meccanismo;
- verificare la resistenza dell'intero sistema ad impatti di lieve entità come succede di frequente durante gli atterraggi;
- controllare che il servomotore riesca a produrre sufficiente coppia per la movimentazione;
- valutare se il sensore TOF è congruente con il modello utilizzato in simulazione;
- verificare l'implementabilità di quanto creato finora e relativi problemi.

5.1 MODIFICHE ALL'HARDWARE

Si sottolinea che, al fine di ospitare i nuovi componenti che si vogliono implementare, la struttura del drone è stata modificata rispetto a quella prevista dal costruttore. In particolare, come già presentato nella sezione 2.4.1, le dimensioni di Raspberry ne vincolano il collocamento dello stesso sulla parte superiore del drone, tramite un apposito adattatore opportunamente progettato. La batteria, quindi, deve necessariamente essere collocata altrove e, in particolare, trova posto solamente sulla zona inferiore del drone. Ciò risulta un vantaggio in quanto, essendo molto pesante, permette di abbassare il baricentro e rendere il velivolo più stabile sia in fase di atterraggio sia genericamente in volo. Questa modifica in parte permette di compensare l'aggiunta delle zampe che, essendo lunghe, tendono a creare un momento d'inerzia non trascurabile. Per il collegamento della batteria si sceglie di adottata la stessa tecnica usata dal costruttore basata su delle fascette con velcro, le quali permettono un agevole smontaggio della batteria quando si necessita di ricaricarla. Proprio per questo, la piastra sottostante del drone (Figura 24), che viene rimpiazzata con una stampata in 3D al fine di ospitare il collegamento dei perni delle zampe, presenta dei fori asolati in grado di ospitare la forma piatta della fascetta.

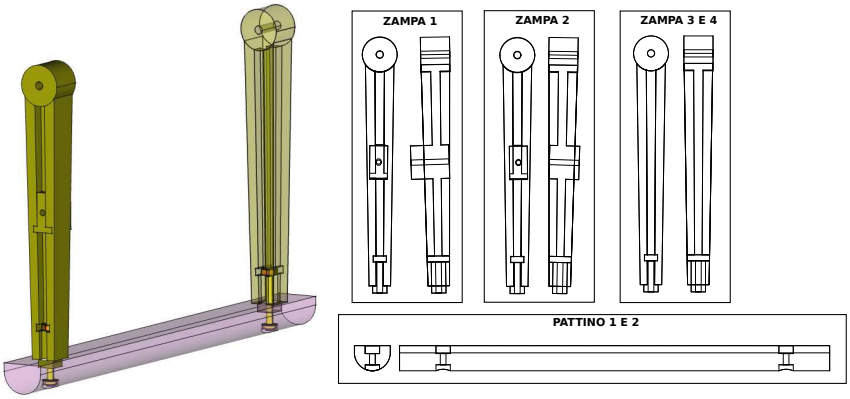

Figura 45: Rendering 3D di una zampa modificata per essere prodotto mediante stampa 3D (a sinistra) e relativo disegno tecnico di ogni singolo pezzo (a destra)

5.1.1 *Adattamento zampe per stampa 3D*

La creazione delle zampe per mezzo della stampa 3D di materiale polimerico a deposizione fusa richiede l'impiego di qualche accorgimento per permettere un accrescimento corretto dei vari strati di materiale. Si rende quindi necessario fare delle modifiche al pezzo al fine di:

- minimizzare il materiale di supporto impiegato, ovvero quel materiale che viene stampato con trama poco densa allo scopo di creare una base d'appoggio al pezzo vero e proprio, come nel caso di concavità nelle parti inferiori o di forme a sbalzo;
- permettere la stampa lungo piani che seguano la logitudinalità delle zampe, in modo da massimizzare le tenuta a flessione ed evitare rotture dei vari strati;
- poter sostituire un pezzo specifico in caso di rottura, evitando di dover stampare tutto l'assieme di grandi dimensioni;

La geometria della zampa viene quindi modificata in modo da permetterne lo smontaggio del pattino creando un collegamento ad incastro a sezione rettangolare (Figura 45). Viene inoltre implementato un sistema di fissaggio a vite dei due pezzi con apposito alloggiamento del dado sull'asta e sede per la vite nella parte inferiore del pattino. Come già previsto, il perno che collega la zampa alla struttura permettendone la rotazione relativa presenta dei fori con sede per la vite, evitando quindi che quest'ultima vada a sfregare durante la rotazione.

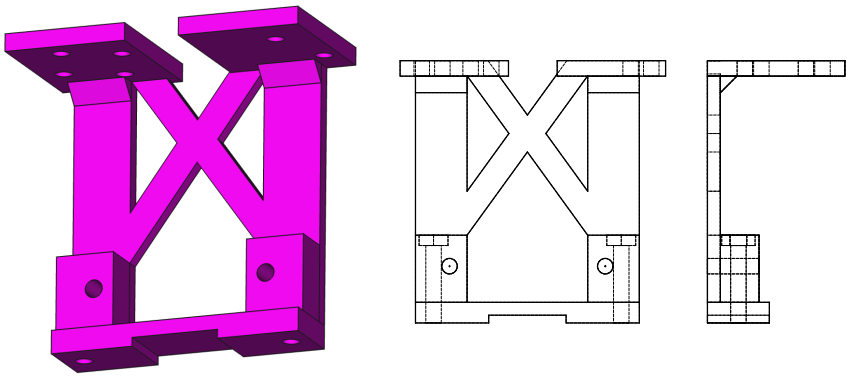

Figura 46: Rendering 3D del supporto del servomotore (a sinistra) e relativo disegno tecnico (a destra)

5.1.2 *Supporto di fissaggio del servomotore*

Per collegare il servomotore saldamente alla struttura del drone si è progettato un apposito supporto, anch'esso stampato in materiale polimerico (Figura 46). Questo va a fissarsi alla piastra sottostante mediante le viti già presenti nel drone che servono per fissare le ali alle due piastre inferiori e mantenere rigida la struttura. Si è scelto di posizionare il servomotore in modo da avere l'asse di rotazione in posizione centrale. Ciò implica che mantenendo il corpo del servomotore, che è a sezione rettangolare, in maniera orizzontale per occupare minor spazio, esso risulti decentrato rispetto alla struttura del drone. Il supporto non è quindi speculare ma risulta leggermente traslato proprio per assecondare questo tipo di scelta. Il posizionamento lungo la direzione longitudinale, invece, è dettato dalla presenza del meccanismo, mentre la distanza dalla piastra è scelta in maniera da posizionare il servomotore inferiormente alla batteria.

Il corpo di questo supporto deve sopportare principalmente sollecitazione di tipo torsionale, dovute alla coppia prodotta dal servomotore. Per questo la forma della struttura presenta due pezzi posti in diagonale a formare una "X" al fine di prevenire la deformazione.

Si è sfruttata la presenza di questo supporto anche per alloggiare il sensore TOF, il quale non deve essere oscurato dalla presenza di alcun componente al di fuori dei pattini delle zampe (il cui oscuramento è già compensato via software). Si è creata quindi una scanalatura della stessa larghezza della PCB ospitante il sensore in modo da bloccarne il movimento. Il TOF scelto, nella sua versione SATEL ovvero adatta per la prototipazione rapida, presenta una PCB di notevoli dimensioni al fine di ospitare comodamente i connettori discreti. Tuttavia, da manuale, è possibile tagliare la scheda lungo il tratto evidenziato dalla fila di fori, riducendone notevolmente le dimensioni ed escludendo la parte che ospita i connettori. La connessione elettrica è garantita tramite delle apposite piedinature facilmente stagnabili alle quali è possibile collegare direttamente i fili.

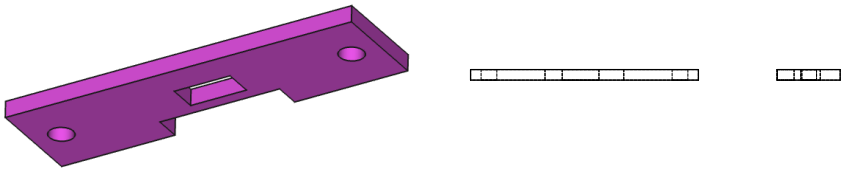

Figura 47: Rendering 3D del pezzo di sostegno del sensore TOF (a sinistra) e relativo disegno tecnico (a destra)

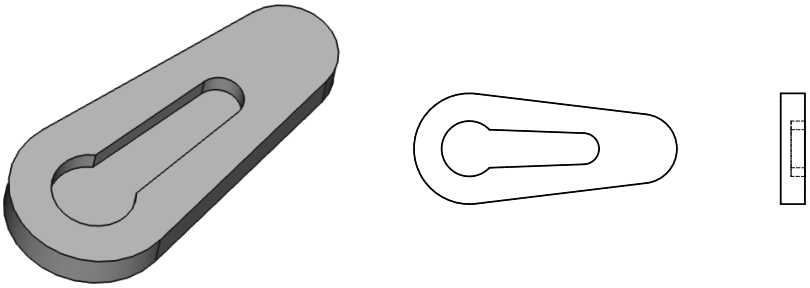

Figura 48: Rendering 3D dell'adattatore del perno del servomotore (a sinistra) e relativo disegno tecnico (a destra)

Un ulteriore pezzo ha il compito di sostenere il TOF e di mantenerlo aderente al supporto del servomotore (Figura 47). Lo scivolamento è prevenuto tramite l'apposita fessura dal quale il sensore TOF vero e proprio (ovvero il pezzo sporgente nero in Figura 16) può uscire, permettendo al sensore di vedere correttamente l'ambiente sottostante. La presenza della rientranza è stata creata per alloggiare le saldature delle connessioni elettriche.

5.1.3 *Adattatore del perno del servomotore*

Sfruttando il perno fornito all'acquisto del servomotore, il quale possiede un'apposita zigrinatura che lo rende solidale all'albero del motore stesso, si aggiunge un ulteriore pezzo che si occupa di adattarne la forma al tratto di meccanismo progettato. Partendo dalla forma del perno del servomotore creato in fase di progettazione, si crea un'alloggiamento della stessa forma del perno del servomotore. I due componenti vengono poi fissati mediante una piccola vite.

5.1.4 *Collegamento del TOF a Raspberry*

La comunicazione tra TOF e Raspberry è garantita mediante il sistema seriale I^2C, molto impiegato nel campo della sensoristica in quanto è facilmente implementabile, economico e permette il collegamento in parallelo di diversi dispositivi ad ognuno dei quali si può assegnare un indirizzo. Raspberry presenta internamente già due bus di trasmissione compatibili con I^2C, utilizzabili mediante i pin presen-

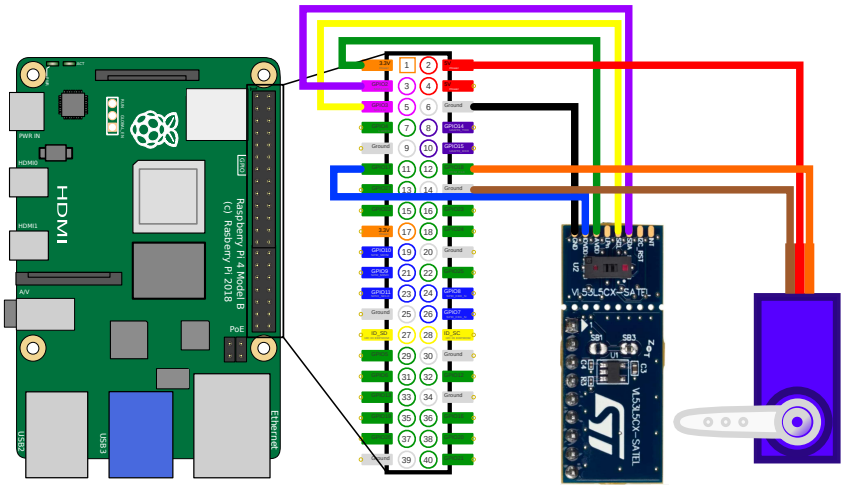

Figura 49: Diagramma di collegamento del sensore VL53L5CX-SATEL e del servomotore a Raspberry Pi 4

ti e di agevolmente comunicazione grazie alle diverse librerie disponibili in rete. Per quanto riguarda il TOF, in particolare, all'interno del sito ufficiale è possibile scaricare delle librerie compatibili con il kernel Linux e ottimizzate proprio per l'uso a bordo di Raspberry. Una serie di codici sorgenti di esempio, in linguaggio C, permettono di inizializzare il sensore e di verificarne la comunicazione.

Il collegamento elettrico tra i due apparati è composto principalmente dalle seguenti linee (Figura 49): [18]

- *INT*: uscita o entrata digitale utilizzabile con linea di interrupt, ma non impiegata.
- *I2C_RST*: collegata a massa tramite una resistenza di 47 $k\Omega$, se portata ad un valore alto e successivamente ad un valore basso effettua il reset del sensore.
- *SDA*: è la linea di trasmissione dati I^2C ed è mantenuta ad un valore alto da un'apposita resistenza di pull-up[1] già presente nella scheda. A Raspberry è collegata al pin *GPIO2*.
- *SCL*: similmente alla precedente, si occupa della trasmissione del clock del collegamento I^2C, impartito dal dispositivo master, che in questo caso è rappresentato da Raspberry il quale si collega tramite il pin GPIO3.

1 Questo tipo di resistenza è utilizzato per mantenere la linea normalmente ad un livello logico alto. Ciascun dispositivo connesso presenta al proprio interno un transistor in grado di cortocircuitare la linea verso massa e quindi portarla al valore logico basso per comunicare. Questa tecnica ha il vantaggio di escludere il conflitto di comunicazione tra dispositivi in quanto nessuno di loro può forzare ad un livello logico alto la linea. Il valore della resistenza è solitamente molto elevato in modo da non generare correnti eccessivamente elevate, ma sufficientemente piccolo da permettere rapidamente la transizione dal valore basso a quello alto.

- *LPn*: attiva o disattiva la modalità LP del TOF, utilizzata per cambiare l'indirizzo I^2C. Avendo un solo dispositivo, non viene collegata.
- *PWRWN*: se portata al valore alto attiva l'integrato che si occupa della conversione della tensione di alimentazione, il quale tuttavia risiede nel pezzo di PCB da eliminare.
- *AVDD*: linea di alimentazione principale a $3,3\ V$ collegato al pin numero 1 di Raspberry.
- *IOVDD*: fornisce l'alimentazione per la comunicazione digitale ed è compatibile con le tensioni $1,8\ V$, $2,8\ V$ e $3,3\ V$. Viene collegato assieme al precedente 0, equivalentemente, al pin 17.
- *GND*: riferimento di massa per tutti i segnali e le alimentazioni, riportato al pin 6 di Raspberry.

5.1.5 *Collegamento del servomotore a Raspberry*

Il servomotore utilizzato, essendo di piccola taglia e quindi richiedente una bassa potenza di alimentazione, può essere collegato direttamente ai pin di Raspberry. In particolare, i tre fili presenti svolgono la seguente funzione (Figura 49):

- filo marrone: massa dell'alimentazione e del segnale, collegabile al pin 14 di Raspberry;
- filo rosso: alimentazione a 5V, la quale viene fornita direttamente da Raspberry mediante il pin 2;
- filo arancione: segnale PWM di comando del servomotore controllato mediante la GPIO18.

Questo servomotore presenta internamente un controllore in grado di comandare la tensione del motore presente all'interno con lo scopo di raggiungere un certo target di posizione nel più breve tempo possibile e di mantenerlo, al limite delle capacità del motore, anche nel caso siano presenti delle coppie esterne. Il target di posizione può essere comunicato variando il periodo di un segnale PWM in maniera lineare rispetto all'orientazione che si vuole ottenere del perno. Un periodo di $1\ ms$ corrisponde alla posizione minima, in questo caso corrispondente a $-90°$, mentre un periodo di $2\ ms$ equivale a $+90°$.

5.2 ADATTAMENTO SOFTWARE

Il software preparato per il controllo del sistema attuato all'interno della simulazione è pensato per poter essere adattato anche all'implementazione reale. In particolare, osservando la Figura 35, il nodo

ROS2 leg_controller.cpp che si occupa di gestire i dati in arrivo dal sensore TOF e controllare conseguentemente il riferimento al servomotore, è costruito in modo da sottoscriversi e pubblicare solamente su topic ROS2, non dovendo mai interfacciarsi direttamente con altre applicazioni. Questa caratteristica permette di renderlo versatile all'utilizzo sia nelle simulazioni sia eseguito a bordo del Raspberry Pi. Lo stesso discorso vale per il nodo di offboard_control.cpp il quale si interfaccia, oltre che con il nodo leg_controller.cpp, con il firmware PX4-Autopilot tramite comunicazione microRTPS, presente anche nel modello reale. Tutti gli altri applicativi, impiegati come plugin di interfaccia al simulatore, devono invece essere adattati per poter comunicare con l'hardware reale e nella fattispecie con il sensore TOF ed il servomotore.

6

CONCLUSIONI

In questa prima parte del più ambizioso progetto QR01 è stata affrontata una prima progettazione di un sistema attuato in grado di consentire l'atterraggio di un quadrirotore su piani inclinati. Inizialmente si sono affrontate delle analisi relative alla scelta del sistema che più si potesse adattare agli obbiettivi, andando a valutare le varie configurazioni possibili in termini di tipo e posizionamento delle zampe, numero e tipo di attuatori, trovando dei parametri dimensionali ottimali che non portino a problemi quali ribaltamento o collisioni. Ciò ha permesso di adottare una struttura composta da due zampe, ovvero una sorta di carrello di atterraggio attuato il cui movimento delle stesse è comandato da un unico servomotore. Il meccanismo che realizza il movimento simultaneo delle due zampe è stato studiato in modo da riuscire ad ottenere gli obbiettivi imposti a priori di inclinazione del piano di atterraggio e ottimizzando la forza necessaria al motore.

Per la verifica del sistema di atterraggio si è preparato un ambiente simulativo su Gazebo, in grado di virtualizzare il firmware del controllore di volo (PX4-Autopilot) e presentante un modello fisico dettagliato del drone. Questo ha implicato uno studio analitico dell'inerzia dei componenti del velivolo per poter riportare un modello affidabile e una calibrazione delle interazioni tra le varie parti. Alcune problematiche si sono verificate in quanto il modello matematico che collega tra loro i pezzi all'interno del simulatore è difficilmente stimabili rispetto al modello reale. Spesso si è preferito perdere in precisione del modello per rendere più realistico il comportamento rispetto a quello aspettato nel mondo reale.

Il sistema di atterraggio è pensato per funzionare senza conoscere le informazioni del piano sottostante e, per questo, si ricava le informazioni tramite un particolare sensore di tipo TOF il quale crea una mappa di profondità 8x8 dello spazio sottostante. Il controllo è basato tramite una serie di nodi e plugin, alcuni dei quali specifici per l'interfacciamento alla simulazione. Tuttavia, il nodo di controllo principale è pensato per essere impiegato anche nel modello reale e si interfaccia agli altri nodi per ricevere ed inviare le informazioni. Quest'ultimo si occupa principalmente di:

- ricezione e pulizia dei dati del TOF;
- interpolazione piano sottostante;
- calcolo della distanza dalla superficie;
- controllo dello yaw per allineare il drone;

L'intero sistema è stato testato andando ad atterrare su 3 piani di inclinazione e orientazione generica, verificandone il corretto comportamento. Tuttavia, in alcuni casi si riscontrano alcune problematiche derivanti da perturbazioni nei calcoli del motore fisico e conseguenti fenomeni oscillatori che però non ne compromettono il risultato finale che risulta conforme agli obbiettivi che ci si è posti.

Il movimento del drone è comandato invece da un ulteriore nodo il quale può essere programmato per seguire le traiettorie volute. Si sono infine realizzati fisicamente i componenti progettati e montati a bordo del drone.

Alcuni miglioramenti che, in futuro, potrebbero essere eseguiti al progetto:

- modellazione accurata dei propulsori nella simulazione;
- messa a punto delle interazioni del modello;
- verifica sperimentale dell'angolo di visione del TOF e creazione di un software per la calibrazione;

La prosecuzione del progetto QR01, tuttavia, potrebbe essere orientata all'ulteriore aumento delle capacità del sistema di atterraggio permettendone l'utilizzo anche in situazioni meno ideali. Inoltre, basandosi su quanto creato, i prossimi step potrebbero puntare verso un ampliamento delle possibilità di interazione del drone, non limitandolo al solo atterraggio ma, ad esempio, andando verso il contatto controllato di oggetti o superfici.

BIBLIOGRAFIA

[1] Shivaji Manik Awchar, Seema P. Diwan, and Pratik Arlikar. Advanced technique for speed control of sensor-less bldc motor. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–5, 2018. doi: 10.1109/ICCUBEA.2018.8697796.

[2] Alistair Forbes. Least-squares best-fit geometric elements, 01 1989.

[3] Davor Kustec. Brushless Motor Kv Rating Explained. https://dronenodes.com/brushless-motor-kv-rating-explained/, n.s. [Online; Aggiornato al 4/03/2022].

[4] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

[5] Steven Peters and John Hsu. Comparison of Rigid Body Dynamic Simulators for Robotic Simulation in Gazebo. https://www.osrfoundation.org/wordpress2/wp-content/uploads/2015/04/roscon2014_scpeters.pdf, n.s. [Online; Aggiornato al 30/04/2022].

[6] Sconosciuto. Holybro Micro Power Module (PM06). https://docs.px4.io/master/en/power_module/holybro_pm06_pixhawk4mini_power_module.html, 2019. [Online; Aggiornato al 3/03/2022].

[7] Sconosciuto. ESCs ∧ Motors. https://docs.px4.io/master/en/peripherals/esc_motors.html, 2021. [Online; Aggiornato al 3/03/2022].

[8] Sconosciuto. BU-501a: Discharge Characteristics of Li-ion. https://batteryuniversity.com/article/bu-501a-discharge-characteristics-of-li-ion, 2021. [Online; Aggiornato al 09/07/2022].

[9] Sconosciuto. Drone technology uses and applications for commercial, industrial and military drones in 2021 and the future. https://www.businessinsider.com/drone-technology-uses-applications?r=US&IR=T, 2021. [Online; Aggiornato al 01/07/2022].

[10] Sconosciuto. Simulation | PX4 User Guide. https://docs.px4.io/v1.12/en/simulation/, 2021. [Online; Aggiornato al 30/04/2022].

[11] Sconosciuto. Pagina GitHub del software BLHeli. https://github.com/bitdump/BLHeli, 2022. [Online; Aggiornato al 4/03/2022].

[12] Sconosciuto. What Is Camera Calibration? https://www.mathworks.com/help/vision/ug/camera-calibration.html, n.s.. [Online; Aggiornato al 17/04/2022].

[13] Sconosciuto. Camera Calibration. https://navigation.ros.org/tutorials/docs/camera_calibration.html, n.s.. [Online; Aggiornato al 17/04/2022].

[14] Sconosciuto. Macro FCInfo. https://wiki.freecadweb.org/Macro_FCInfo, n.s.. [Online; Aggiornato al 25/04/2022].

[15] Sconosciuto. About Gazebo. https://gazebosim.org/about, n.s.. [Online; Aggiornato al 30/04/2022].

[16] Sconosciuto. RTPS/DDS Interface: PX4-Fast RTPS(DDS) Bridge. https://docs.px4.io/v1.12/en/middleware/micrortps.html, n.s.. [Online; Aggiornato al 05/06/2022].

[17] Sconosciuto. SDFormat Specification. http://sdformat.org/spec, n.s.. [Online; Aggiornato al 30/04/2022].

[18] Sconosciuto. Breakout Boards for VL53L5CX. https://www.st.com/en/evaluation-tools/vl53l5cx-satel.html, n.s.. [Online; Aggiornato al 03/07/2022].

[19] Sconosciuto. Time-of-Flight sensors. https://www.st.com/en/imaging-and-photonics-solutions/time-of-flight-sensors.html, n.s.. [Online; Aggiornato al 02/05/2022].

[20] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2010. ISBN 9781846286414. URL https://books.google.it/books?id=jPCAFmE-logC.