

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Generazione e Inseguimento di Traiettorie Virtuali per un Quadricottero Vincolato: Implementazione e Verifica Sperimentale del Sistema di Controllo

Generation and Tracking of Virtual Trajectories for a Constrained Quadrotor: Implementation and Experimental Validation of the Control System

Candidato:
Lorenzo Di Stefano

Relatore:
Prof. Andrea Bonci

Anno Accademico 2025-2026

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

A chi ha sempre
creduto in me

Abstract

L'obiettivo principale di questo progetto consiste nello sviluppo di una strategia di controllo in retroazione per la guida di un drone quadricottero lungo una traiettoria predefinita in ambiente simulato. Il velivolo è vincolato a un supporto, che ne consente la rotazione ma non la traslazione, di conseguenza, le prove non hanno previsto il volo libero del drone, ma hanno riguardato esclusivamente una simulazione basata su traiettorie virtuali.

Il sistema di controllo si basa su uno schema di regolazione in cascata a due gradi di libertà, in particolare, il controllore esterno, che sfrutta un modello di controllo del secondo ordine, ha il compito di confrontare istante per istante la traiettoria di riferimento con la posizione virtuale del velivolo, calcolando gli angoli richiesti di pitch e roll, da inviare in seguito come riferimenti ai controllori interni.

L'intera traiettoria viene generata in MATLAB e trasmessa alla scheda STM32 attraverso un protocollo di comunicazione seriale dedicato. Da quel momento tutte le operazioni sono gestite autonomamente dal microcontrollore, tramite una macchina a stati che gestisce l'armamento, l'hovering, l'inseguimento della traiettoria e il disarmo finale.

Il progetto prende come punto di partenza il lavoro precedente effettuato sul drone, di cui vengono riportati in seguito alcuni aspetti rilevanti. La prima parte della tesi fornisce un'introduzione al concetto di UAV, con riferimento alla struttura costitutiva e al funzionamento fisico che ne consente la dinamica di volo, insieme alla trattazione dei modelli matematici impiegati per eseguirne il controllo. Nella seconda parte della tesi viene descritta la strumentazione hardware utilizzata durante il corso del progetto insieme alle soluzioni progettuali e alle scelte di implementazione effettuate. Infine, nella terza parte della tesi vengono descritte tutte le operazioni effettuate dal lato software, insieme ai rispettivi test effettuati e risultati ottenuti durante lo svolgimento del progetto.

Indice

Abstract iv

Indice v

1. Introduzione al Progetto 1

1.1. Introduzione agli UAV . . . 1

1.2. Dinamica del Volo e Problematiche di Controllo . . . 3

1.3. Introduzione al Problema del Controllo . . . 5

1.4. Evoluzione della Logica di Controllo . . . 6

1.5. Presentazione del Progetto . . . 9

2. Modello Matematico del Sistema 11

2.1. Generalità sulla struttura del quadricottero . . . 11

2.2. Fisica del Quadricottero . . . 12

2.2.1. Derivazione delle Grandezze Fisiche Fondamentali . . . 13

2.2.2. Matrici di Rotazione . . . 14

2.2.3. Derivazione della Matrice T . . . 15

2.2.4. Roll, Pitch e Yaw . . . 16

2.2.5. Spinta . . . 18

2.2.6. Forze e Momenti Agenti sul Quadricottero . . . 18

2.3. Modello di Controllo . . . 21

2.3.1. Calcolo delle Costanti . . . 22

2.4. Modello Matematico della Traiettoria 23
2.4.1. Relazioni tra Angoli e Accelerazioni 23
2.4.2. Traiettoria a Strappo Minimo 24
2.4.3. Applicazione del Modello al Sistema 26
2.5. Modello del Controllo a Ciclo Chiuso 27

3. Strumentazione Hardware 29

3.1. Scheda STM32 H745 Nucleo-144 29
3.2. Motori 30
3.3. ESC 32
3.4. Batteria 33
3.5. Eliche 34
3.6. IMU 35
3.7. PDB 36
3.8. Radiocomando 36
3.9. Schema dei Collegamenti 37
3.10. Prova dei Motori 38
3.10.1. Calibrazione ESC 39

4. Architettura Implementativa 41

4.1. Accenni allo Sviluppo dello Yaw 41
4.2. Ricezione dei Dati da MATLAB 43
4.3. Utilizzo della Traiettoria a Jerk Minimo 44
4.4. Implementazione della Macchina a Stati Finiti 45
4.5. Controllo in Catena Chiusa 47

5. Sviluppo e Simulazioni Software 49

5.1. Comunicazione tra MATLAB e Microcontrollore 49
5.1.1. Collegamento Fisico e Parametri della Linea 49
5.1.2. Scelta del Protocollo Testuale 49

5.1.3. Ricezione a Interrupt della Coda di Righe 50
5.1.4. Elaborazione delle Stringhe 51
5.1.5. Primo formato dei messaggi: invio in tempo reale 51

5.2. Conversione della Traiettoria in Angoli di Riferimento 52
5.2.1. Profilo a Jerk Minimo 52
5.2.2. Passaggio alle Traiettorie Continue 53
5.2.3. Utilizzo del Modello Inverso 55

5.3. Trasmissione della Telemetria dalla Scheda 55
5.3.1. Utilizzo di `printf` sulla Seriale 55
5.3.2. Frequenza e contenuto 56
5.3.3. Tempo di Trasmissione e Campionamento Effettivo 56
5.3.4. Lettura dei Dati in MATLAB 57

5.4. Stima della Posizione del Drone 57
5.4.1. Modello 57
5.4.2. Compensazione dell'Offset 57
5.4.3. Integrazione 58
5.4.4. Ricostruzione della Traiettoria 58

5.5. Controllo in Posizione a Ciclo Chiuso 59
5.5.1. Caricamento Anticipato della Traiettoria 59
5.5.2. Macchina a Stati 60
5.5.3. Legge di Controllo di Posizione 60
5.5.4. Scelta dei Parametri 61

5.6. Sintesi del Controllore PID 62
5.6.1. Funzionamento del PID 62
5.6.2. Tuning del Controllore 63
5.6.3. Discretizzazione del PID 64
5.6.4. Modifiche Attuate 66

6. Test e risultati **69**

6.1. Condizioni di Partenza 69

6.2. Primo Tentativo Invio Riferimenti da MATLAB 71

6.3. Correzione Invio Riferimenti da MATLAB 72

6.4. Assetto da MATLAB in Catena Aperta 73
6.5. Traiettoria da MATLAB in Catena Aperta 76
6.6. Circonferenza da MATLAB in Catena Chiusa 77
6.7. Rettilineo da MATLAB in Catena Chiusa. 78
6.8. Triangolare da MATLAB in Catena Chiusa 79

Conclusioni e Sviluppi Futuri **81**

Bibliografia **83**

1

Introduzione al Progetto

In questo primo capitolo viene introdotto il contesto generale del progetto di tesi, trattando inizialmente le generalità sul concetto di veivolo a pilotaggio remoto (UAV) e successivamente le relative architetture di controllo del volo. Su tale base teorica si basa la presentazione del caso di studio specifico affrontato nel lavoro, che verrà approfondita alla fine del capitolo.

1.1. Introduzione agli UAV

Con il termine *drone* si indica comunemente un velivolo privo di pilota a bordo, la cui denominazione tecnica è UAV (*Unmanned Aerial Vehicle*) o, secondo la terminologia ICAO ed EASA, RPAS (*Remotely Piloted Aircraft System*). Più correttamente si parla di UAS (*Unmanned Aircraft System*), poiché il velivolo costituisce solamente una parte di un sistema che comprende anche la stazione di terra, il collegamento radio e il carico utile, oltre agli eventuali sistemi di supporto necessari al suo funzionamento [20].

L'idea di un velivolo senza equipaggio compare già durante la Prima Guerra Mondiale, quando furono sperimentati ordigni volanti a guida automatica, e negli anni Trenta il britannico DH.82B *Queen Bee*, un bersaglio radiocomandato, diede origine per assonanza

al termine *drone*. Per decenni lo sviluppo è rimasto confinato all'ambito militare, dove gli UAV sono stati impiegati principalmente per missioni di ricognizione e sorveglianza. La svolta verso l'impiego civile è avvenuta a partire dagli anni Duemila, a causa di vari fattori tecnologici, tra i quali la miniaturizzazione dei sensori inerziali MEMS, che ha reso disponibili accelerometri e giroscopi a basso costo e basso consumo; la diffusione delle batterie ai polimeri di litio e dei motori brushless, con rapporti tra potenza e peso prima impensabili; e la disponibilità di microcontrollori capaci di eseguire algoritmi di stima e controllo in tempo reale a bordo di varie piattaforme [20].

Nota 1.1 Gli UAV vengono classificati secondo diversi criteri, in funzione delle loro caratteristiche operative. Dal punto di vista della configurazione aerodinamica si distinguono velivoli ad ala fissa, ad ala rotante e configurazioni ibride (o convertiplano). Invece, dal punto di vista dimensionale e prestazionale si va dai *micro air vehicles* (MAV) di pochi centimetri ai sistemi MALE (*Medium Altitude Long Endurance*) e HALE (*High Altitude Long Endurance*), con apertura alare di decine di metri e autonomia di giorni [20]. I velivoli ad ala fissa offrono maggiore efficienza aerodinamica e quindi una maggiore autonomia, ma richiedono una velocità minima di sostentamento e non sono in grado di rimanere fermi in un punto. Al contrario, i velivoli ad ala rotante possiedono capacità di decollo e atterraggio verticale (VTOL) e di volo stazionario, al costo di un'autonomia molto inferiore.

Nella famiglia dei velivoli ad ala rotante, la configurazione che si è imposta con maggiore decisione nell'ultimo decennio è il **quadricottero** (o *quadrotor*), costituito da quattro rotori a passo fisso disposti simmetricamente rispetto al centro di massa, azionati indipendentemente e controrotanti a coppie per compensare la coppia di reazione complessiva. L'idea di drone quadricottero compare già nel 1907 con il *Gyroplane No. 1* dei fratelli Bréguet e nel 1922 con la macchina di Oehmichen, che però risultavano ingovernabili con le tecnologie dell'epoca proprio per l'impossibilità di stabilizzare manualmente il velivolo [7].

Il successo attuale del quadricottero è dovuto alla sua semplicità meccanica, siccome, a differenza dell'elicottero convenzionale, che richiede un piatto oscillante e rotori a passo variabile, il quadricottero riesce a muoversi agendo unicamente sulla velocità di rotazione dei quattro motori. Un incremento uniforme delle quattro velocità produce una variazione della spinta totale e quindi del moto verticale, mentre una variazione di spinta fra coppie

di rotori genera i momenti di rollio, beccheggio e imbardata [7, 15]. Ne conseguono costi contenuti, elevata manovrabilità, buona scalabilità dimensionale e assenza di parti meccaniche critiche. Per quanto riguarda le limitazioni, si hanno una scarsa efficienza energetica, un carico utile limitato e una marcata sensibilità ai disturbi atmosferici. Questa piattaforma costituisce oggi la base per diverse applicazioni civili quali il rilievo fotogrammetrico, l'ispezione di infrastrutture, l'agricoltura di precisione, la ricerca e il soccorso, la logistica leggera e la ripresa cinematografica [20], ambiti la cui crescita è accompagnata da un quadro normativo in rapida evoluzione, in Europa definito dai Regolamenti (UE) 2019/947 e 2019/945.

1.2. Dinamica del Volo e Problematiche di Controllo

Dal punto di vista sistemistico, la semplicità meccanica del quadricottero si traduce in una notevole complessità del sistema di controllo. Il velivolo è un corpo rigido con sei gradi di libertà, di cui tre traslazionali e tre rotazionali, i cui ingressi di controllo possono essere espressi in termini delle velocità dei quattro rotori, oppure, in maniera equivalente, attraverso la spinta totale e i tre momenti. Il sistema viene definito **sottoattuato**, ovvero, non è possibile assegnare arbitrariamente e in modo indipendente posizione e assetto [7, 15].

Da questa proprietà deriva l'accoppiamento fondamentale fra assetto e traslazione. Poiché la spinta dei rotori è diretta lungo l'asse verticale del corpo, l'unico modo per generare una forza orizzontale è inclinare l'intero velivolo. Il moto lungo una direzione del piano è quindi ottenuto indirettamente, attraverso una variazione di assetto. Il problema di far seguire al drone una traiettoria assegnata non si riduce perciò al semplice inseguimento di un riferimento di posizione, ma richiede di generare in modo coordinato un profilo di assetto compatibile con le accelerazioni desiderate.

A ciò si aggiungono ulteriori difficoltà. La dinamica del quadricottero è **intrinsecamente non lineare**, poichè le equazioni del moto contengono i termini di rotazione della matrice di assetto, e dipendono quadraticamente dalla velocità dei rotori. Il sistema è inoltre **instabile in anello aperto**, siccome, in assenza di un'azione stabilizzante, eseguita a frequenze tipicamente dell'ordine di centinaia di Hz, il velivolo diverge rapidamente dalla

condizione di equilibrio. Il controllo non costituisce quindi un semplice miglioramento delle prestazioni, ma una condizione necessaria per la volabilità del sistema.

Nel caso considerato, il modello matematico è inoltre affetto da incertezze non trascurabili, tra le quali momenti d'inerzia, coefficienti di spinta e coppia noti con approssimazione, la tensione della batteria incostante durante il volo che modifica il guadagno degli attuatori, e fenomeni aerodinamici quali l'effetto suolo, il *blade flapping* e la resistenza indotta dalla velocità di traslazione. Infine, lo stato del velivolo non è misurato direttamente ma stimato a partire da sensori rumorosi e affetti da deriva (IMU, barometro, GNSS, sistemi di visione), il che richiede algoritmi di fusione sensoriale e introduce ritardi ed errori di stima nell'anello di retroazione [15].

Nota 1.2 È utile distinguere tre formulazioni del problema di controllo di un drone. Nella **regolazione di punto** (*set-point regulation*) si chiede al velivolo di raggiungere e mantenere una posizione assegnata; nell'**inseguimento di traiettoria** (*trajectory tracking*) il riferimento è una funzione del tempo, e si richiede che il velivolo si trovi in un punto specificato in un istante di tempo specificato; nell'**inseguimento di percorso** (*path following*) è assegnata solo la curva geometrica, mentre la legge oraria resta libera [16]. Il presente lavoro rientra nella seconda categoria.

La soluzione utilizzata per affrontare tali difficoltà sfrutta la differenza di velocità fra la dinamica rotazionale, più rapida, e quella traslazionale, più lenta. Ne deriva una struttura di **controllo in cascata**, con un anello esterno di posizione che, a partire dall'errore rispetto alla traiettoria desiderata, produce i riferimenti di assetto e di spinta, e un anello interno di assetto, eseguito a frequenza più elevata, che li insegue agendo sui singoli rotori [7, 15]. La validità teorica di questo schema deriva dalla proprietà di **piattezza differenziale** del modello del quadricottero, che permette di calcolare lo stato e gli ingressi del sistema in funzione delle sole coordinate di posizione, dell'angolo di imbardata e delle loro derivate. Quindi, a partire da una traiettoria sufficientemente regolare è possibile ricavare analiticamente i riferimenti per tutte le variabili del sistema [16].

1.3. Introduzione al Problema del Controllo

Prima di esaminare le varie tecniche utilizzabili, conviene richiamare in termini generali il concetto di problema di controllo e gli elementi fondamentali che lo caratterizzano. Si consideri un sistema dinamico, detto *impianto* o *processo*, sul quale è possibile agire mediante una o più variabili di ingresso e del quale si osservano una o più variabili di uscita. Sul sistema agiscono inoltre grandezze non manipolabili, chiamate *disturbi*. Controllare il sistema significa determinare l'andamento degli ingressi affinché le uscite assumano un comportamento desiderato, descritto da un segnale di *riferimento*, nonostante la presenza dei disturbi e l'incertezza sul modello [2].

Nel **controllo in anello aperto** l'ingresso viene calcolato solamente sulla base del riferimento e di un modello del processo, senza misurare l'effetto prodotto. Questa soluzione è semplice ed economica, ma funziona soltanto se il modello è accurato e i disturbi sono trascurabili. Qualsiasi divario si traduce in un errore che il sistema non è in grado di rilevare, e un processo instabile non può in alcun modo essere stabilizzato. Nel **controllo in retroazione** (*feedback*), invece, l'uscita viene misurata e confrontata con il riferimento, e la differenza, detta *errore*, alimenta il controllore, che genera l'azione correttiva. È questo meccanismo che conferisce al sistema controllato la capacità di stabilizzare processi instabili, di attenuare l'effetto dei disturbi e di rendere il comportamento complessivo poco sensibile alle variazioni dei parametri dell'impianto [2].

Nota 1.3 Nonostante ciò, l'impiego della retroazione può comunque comportare alcune controindicazioni. L'anello di controllo richiede infatti la disponibilità di sensori, ne immette il rumore di misura nell'anello di controllo e, soprattutto, può rendere instabile un sistema che in anello aperto era stabile, siccome un guadagno troppo elevato o un ritardo eccessivo fanno sì che l'azione correttiva arrivi in ritardo di fase e si sommi all'errore anziché annullarlo.

Le specifiche di progetto di un sistema di controllo riguardano tipicamente quattro aspetti. La **stabilità** è il requisito primario, e richiede che in corrispondenza di ingressi e condizioni iniziali limitate le variabili del sistema rimangano limitate. Le **prestazioni in transitorio** descrivono la qualità della risposta e si esprimono mediante indici quali il tempo di salita, la sovraelongazione massima e il tempo di assestamento. La **precisione**

a regime nella risposta al termine del regime transitorio. Il principio del modello interno stabilisce che l'annullamento di tale errore richiede che il controllore contenga al proprio interno un modello del segnale di riferimento da inseguire oppure un modello del disturbo da annullare, in modo che il controllore possa calcolare la risposta sulla base di un segnale già noto. La **robustezza**, infine, misura la capacità del sistema di conservare stabilità e prestazioni al variare dei parametri dell'impianto rispetto al modello nominale, e viene quantificata tramite i margini di guadagno e di fase, oppure mediante le norme delle funzioni di sensitività [2].

Nota 1.4 Tali requisiti non sono indipendenti né migliorabili a causa di specifiche limitazioni strutturali, quali la relazione algebrica che lega la funzione di sensitività a quella di sensitività complementare, la quale impone che l'attenuazione dei disturbi su un intervallo di frequenze comporti amplificazione in un altro, e i vincoli imposti da ritardi, zeri a fase non minima e saturazione degli attuatori. Il progetto di un controllore è quindi sempre un compromesso.

Il punto di partenza del progetto del controllore è il **modello matematico** del processo, ottenibile per via analitica a partire dalle leggi fisiche, per via sperimentale mediante tecniche di identificazione, o con approcci misti. Il modello può essere espresso come funzione di trasferimento, nel caso di sistemi lineari e stazionari, oppure in forma di **spazio di stato**, che si estende ai sistemi non lineari, tempo-varianti e multivariabili. Poiché i controllori sono oggi implementati su calcolatore elettronico, a valle del progetto occorre considerare anche gli effetti della discretizzazione, della quantizzazione e della scelta del periodo di campionamento [2].

1.4. Evoluzione della Logica di Controllo

La teoria del controllo è una disciplina relativamente recente nella sua formulazione teorica, sebbene le sue applicazioni pratiche abbiano origini molto antiche. Ad esempio, i regolatori di livello a galleggiante degli orologi ad acqua di Ktesibios, nel III secolo a.C., sono una realizzazione di un anello di retroazione. Il dispositivo generalmente indicato come predecessore del controllo industriale è il **regolatore centrifugo** applicato da James Watt alla macchina a vapore nel 1788, che regola automaticamente l'ammissione del vapore in funzione della velocità dell'albero. Per quasi un secolo, tuttavia, questi

dispositivi furono progettati principalmente su base empirica, fino a quando i fenomeni oscillatori osservati in alcune installazioni resero necessaria un'analisi più rigorosa. Nel 1868, **James Clerk Maxwell**, nella memoria *On Governors*, linearizzò le equazioni del regolatore e ricondusse lo studio della stabilità all'analisi delle radici dell'equazione caratteristica. Il problema algebrico che ne derivava fu risolto in forma generale da Routh e, indipendentemente, da Hurwitz [3].

Il periodo più rilevante per quanto riguarda il **controllo classico** è compreso fra le due guerre mondiali. Nel 1922 **Nicolas Minorsky**, studiando il governo automatico del timone delle navi, propose e analizzò una legge di controllo che comprende le tre azioni proporzionale, integrale e derivativa, fornendo di fatto la prima formulazione del regolatore PID [3]. Nei laboratori Bell, l'invenzione dell'amplificatore in retroazione negativa da parte di Harold Black condusse **Harry Nyquist** a formulare nel 1932 il criterio di stabilità in frequenza [18] e **Hendrik Bode** a sviluppare negli anni successivi gli strumenti di analisi in ampiezza e fase e il concetto di margine di stabilità [4]. Nel 1942 **Ziegler e Nichols** pubblicarono le celebri regole empiriche di taratura dei regolatori PID a partire da semplici prove sperimentali [1], mentre la Seconda Guerra Mondiale, con i problemi di puntamento automatico dell'artiglieria contraerea, condusse allo sviluppo diffuso della teoria dei servomeccanismi presso il Radiation Laboratory del MIT. A completare il quadro contribuirono il metodo del luogo delle radici di Evans (1948) e la teoria della stima di Wiener.

L'avvento del calcolatore digitale e le esigenze poste dalla corsa allo spazio aprirono negli anni Sessanta il periodo del **controllo moderno**, caratterizzato dall'abbandono della descrizione in frequenza in favore della rappresentazione in spazio di stato. **Rudolf Kálmán** introdusse i concetti di raggiungibilità e osservabilità, la formulazione del regolatore lineare quadratico (LQR) e, nel 1960, il filtro che porta il suo nome [2, 12]; la combinazione dei due nel controllo LQG (*Linear Quadratic Gaussian*) trovò applicazione immediata, fra l'altro, nel sistema di guida del programma Apollo. Parallelamente si affermò il **controllo digitale**, favorito dalla diffusione dei microprocessori a partire dagli anni Settanta.

La constatazione che il controllo LQG poteva presentare margini di robustezza arbitrariamente ridotti in presenza di incertezze di modello stimolò, negli anni Ottanta, lo

sviluppo del **controllo robusto**: la formulazione H_∞ proposta da Zames nel 1981 [21] e le successive tecniche di μ-sintesi affrontano esplicitamente il progetto a fronte di una famiglia di modelli anziché di un modello nominale. Nello stesso periodo maturarono il **controllo adattativo**, in cui i parametri del regolatore vengono aggiornati in linea sulla base dei dati misurati, e l'insieme delle tecniche di **controllo non lineare** basate sulla teoria di Lyapunov, quali la linearizzazione per retroazione, il *backstepping* e il controllo a struttura variabile con modi scorrevoli [13]. Di origine industriale è invece il **controllo predittivo** (MPC), nato negli impianti petrolchimici e oggi diffuso anche in applicazioni rapide grazie alla potenza di calcolo disponibile, che determina l'azione di controllo risolvendo a ogni passo un problema di ottimizzazione su un orizzonte finito, con il vantaggio di gestire esplicitamente i vincoli [8]. Più recentemente si sono affermati approcci basati sull'apprendimento e sull'uso diretto dei dati, particolarmente studiati in ambito aeronautico e robotico.

Nonostante la varietà di strumenti disponibili, si stima che la grande maggioranza degli anelli di controllo industriali sia ancora di tipo **PID** [1]. Tra i vari motivi si ha che il regolatore PID richiede la taratura di soli tre parametri, ovvero l'azione proporzionale, che determina la prontezza della risposta all'errore istantaneo; l'azione integrale, che accumula l'errore passato e garantisce l'annullamento dell'errore a regime; e l'azione derivativa, che anticipa l'andamento futuro dell'errore introducendo un effetto smorzante. Il progetto di un regolatore PID non richiede un modello accurato del processo, poichè esistono metodi di taratura sperimentali consolidati, inoltre il costo computazionale è minimo, il che lo rende ideale per implementazioni su microcontrollore, e presenta ottime prestazioni nelle architetture in cascata.

Per queste ragioni, il regolatore PID rappresenta la soluzione adottata nella quasi totalità dei controllori di volo per multirotori, inclusi i progetti *open source* più diffusi quali PX4 e ArduPilot. Inoltre, studi comparativi condotti su quadricotteri reali hanno mostrato prestazioni paragnoabili a quelle di tecniche di controllo più sofisticate, a fronte di una complessità realizzativa sensibilmente inferiore [7]. Su queste considerazioni si fonda la scelta effettuata nel presente lavoro di tesi, di cui si darà conto nel dettaglio nei capitoli successivi, dove verranno presentate la formulazione analitica del regolatore, le sue varianti implementative e i criteri di taratura adottati.

1.5. Presentazione del Progetto

Dopo aver introdotto nei paragrafi precedenti i principali aspetti teorici e storici riguardanti il controllo di un drone, si passa ora agli aspetti applicativi del progetto. L'obiettivo del presente lavoro consiste nello sviluppo e nell'implementazione di un'architettura di controllo in retroazione per un quadricottero, finalizzata all'inseguimento di traiettorie virtuali desiderate nel piano orizzontale. Il velivolo che è stato impiegato nei test di volo è vincolato da cavi di sicurezza, per cui le prove sono state condotte in condizioni controllate e senza spostamenti effettivi del baricentro.

Per il controllo di volo si è fatto impiego di un microcontrollore STM32H745 (core Cortex-M4), che aquisisce gli angoli di assetto da un sensore inerziale BNO055 attraverso il bus di sistema I2C, genera i segnali PWM per i quattro motori e riceve in ingresso il segnale del radiocomando, mantenuto come dispositivo di arresto di emergenza, insieme al segnale ricevuto da MATLAB, attraverso un collegamento seriale con protocollo di comunicazione testuale.

Il progetto prende come punto di partenza un lavoro precedente [11], dedicato al controllo tramite radiocomando del velivolo sugli assi di pitch e roll, seguito da un progetto intermedio sul controllo dell'asse di yaw. Da questi progetti sono stati ereditati il codice del microcontrollore e i regolatori PID di assetto già tarati, comandati esclusivamente dai riferimenti del radiocomando. A questa base il presente lavoro ha aggiunto la comunicazione seriale con MATLAB e la gestione remota dei riferimenti, la generazione di traiettorie a jerk minimo, la stima della posizione e l'anello di controllo esterno a bordo della scheda, l'esecuzione automatica della sequenza di manovra e i codici MATLAB per l'acquisizione e l'analisi dei dati sperimentali.

L'architettura di controllo impiegata nel sistema è in cascata. L'anello interno è composto da tre regolatori PID, ciascuno deputato al controllo di uno dei tre angoli di assetto del velivolo (rollio, beccheggio e imbardata). L'anello esterno è invece descritto da una legge proporzionale-derivativa del secondo ordine, e viene utilizzato per generare i riferimento di assetto a partire dall'errore tra la posizione di riferimento e quella stimata. Sia l'esecuzione dell'anello interno sia la stima della posizione, basata sulla misura angolare fornita

dal sensore inerziale, avvengono all'interno della scheda. La difficoltà principale del lavoro ha riguardato dunque la dinamica del controllo in posizione, siccome, in assenza di un sensore dedicato, la posizione è stata ricavata per doppia integrazione, ed è quindi soggetta a deriva. I limiti e le conseguenze di questa scelta verranno discussi in seguito.

Il lavoro si è sviluppato per fasi successive. In una prima fase MATLAB calcolava le traiettorie, le convertiva in angoli di riferimento e le inviava alla scheda in tempo reale. In una seconda fase la traiettoria è stata trasferita per intero al microcontrollore prima del volo, il quale ha assunto autonomamente la gestione della sequenza di manovra e il calcolo dei riferimenti. Nella versione finale del sistema, il ruolo di MATLAB si limita all'invio, prima del volo, del riferimento di posizione. Una volta che la scheda lo ha memorizzato, il controllore proporzionale-derivativo a bordo elabora tale riferimento e genera gli angoli di assetto desiderati, che costituiscono a loro volta i riferimenti per i regolatori PID dell'anello interno. Questi ultimi agiscono infine sulla la velocità di rotazione dei quattro motori, e il quadricottero procede a spostarsi di conseguenza.

I capitoli successivi illustrano nel dettaglio l'implementazione dell'architettura di controllo, partendo dal modello matematico alla base del progetto, fino ad arrivare alle fasi di sviluppo che hanno portato alla versione definitiva e le motivazioni delle scelte compiute lungo il percorso.

2

Modello Matematico del Sistema

Un modello matematico descrive il comportamento di un sistema fisico mediante un insieme di equazioni. In questo capitolo verranno derivati il modello dinamico del quadricottero e i metodi impiegati per il controllo del volo e il calcolo della traiettoria.

2.1. Generalità sulla struttura del quadricottero

Un drone quadricottero è un aeromobile dotato di quattro eliche orizzontali equamente distanziate dal corpo centrale. Le generazioni più moderne sono progettate per volare utilizzando un sistema di controllo e dei sensori che permettono il volo sia in ambienti interni che esterni. Nel caso di questo progetto il drone non è stato posto in condizione di volo libero, ma è rimasto legato ad un supporto, in modo tale da permettergli esclusivamente l'inclinazione richiesta.

Il quadricottero ha una struttura simmetrica. Ogni braccio ospita un motore e un'elica, i quali sono incaricati di garantire la spinta e la stabilità del dispositivo. Il corpo centrale contiene l'elettronica di controllo, che è composta da:

- Una **PDB (Power Distribution Board)**, scheda elettronica progettata per distribuire l'alimentazione elettrica in modo efficiente e sicuro a tutti i componenti

del veicolo;

- Un **sensore IMU**, che misura e fornisce informazioni riguardo all'orientamento;
- **Scheda STM32 H745 NUCLEO-144**, posta nella parte superiore del drone;
- Quattro **ESC**, uno per ogni braccio, collegati al controllore STM, per gestire la velocità del relativo motore collegato

La struttura è progettata per essere leggera in modo da garantire la possibilità di volare, ma allo stesso tempo robusta per garantire maggior maneggevolezza e resistenza ad eventuali urti [5].

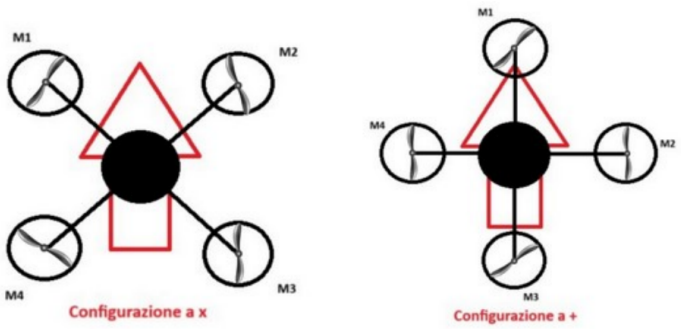


Figura 2.1. Configurazioni del drone

Nota 2.1 Il drone può essere disposto in configurazione a "+" (più) o a "x" (croce). Queste due configurazioni differiscono per il diverso orientamento rispetto al corpo del drone dei vettori che individuano il sistema di riferimento solidale con esso e nella diversa distribuzione delle forze che ciascun motore deve esercitare per effettuare movimenti lungo uno qualunque dei suoi gradi di libertà, ma che non ne cambia le caratteristiche complessive di volo. Entrambe le configurazioni garantiscono comunque una perfetta simmetria del drone, che assieme alla struttura in genere molto compatta rende il drone un sistema adatto allo stabilizzarsi in un punto dello spazio durante il volo (hovering). Nel caso in esame è stata adottata la configurazione a croce.

2.2. Fisica del Quadricottero

Adesso che abbiamo introdotto la struttura del quadricottero, possiamo iniziare a descrivere tutti i principi fisici sopra i quali si basa il volo del drone.

2.2.1. Derivazione delle Grandezze Fisiche Fondamentali

Un quadricottero può essere considerato un sistema fisico dotato di sei gradi di libertà, di cui tre direzioni di traslazione (assi X, Y, Z), che descrivono lo **spostamento lineare** del veicolo e tre direzioni di rotazione (angoli ϕ, θ,ψ), che descrivono lo **spostamento angolare**. Per la descrizione dell'assetto e della posizione del drone e per la definizione delle equazioni che descrivono il movimento del veicolo, introduciamo due sistemi di riferimento

$$O_i = [\ x \quad y \quad z\ ] \qquad O_b = [\ x_b \quad y_b \quad z_b\ ]$$

Il primo è il **sistema di riferimento inerziale** (detto Earth fixed frame), ovvero il sistema solidale al suolo, mentre il secondo è il **sistema di riferimento solidale al veicolo** (detto Body Fixed Frame), che ha come origine il centro di massa del veicolo [9]. I due sistemi di riferimento vengono rappresentati nella Figura 2.2

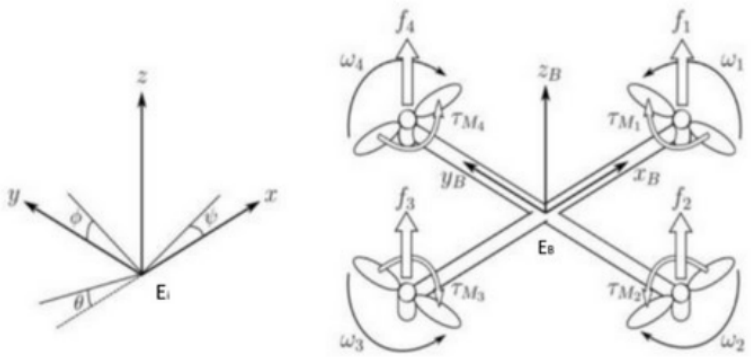


Figura 2.2. Sistema inerziale assoluto e sistema inerziale solidale al telaio del drone

In figura vengono anche rappresentati l'angolo di roll (o rollio) ϕ, l'angolo di pitch (o beccheggio) θ e l'angolo di yaw (o imbardata) ϕ. Inoltre, f_i rappresenta la forza di spinta del motore i-esimo, mentre ω_i ed M_i ne rappresentano rispettivamente la velocità angolare ed il momento angolare.

Per i calcoli riguardanti la dinamica del drone, faremo riferimento alla posizione assoluta del drone con ε e con η il suo assetto nel sistema inerziale O_i. Nell'assetto è possibile indicare l'angolo di roll ϕ (rotazione intorno all'asse x), l'angolo di pitch θ (rotazione intorno all'asse y) e l'angolo di yaw ψ (rotazione attorno all'asse z). Otteniamo così:

$$\varepsilon = [\ x \quad y \quad z\ ] \qquad \eta = [\ \phi \quad \theta \quad \psi\ ]$$

Inoltre, indichiamo con v_b il vettore delle velocità lineari nel sistema di riferimento solidale al drone, e con ω_b quello delle velocità angolari nel sistema solidale. I due vettori possiedono le seguenti componenti:

$$v_b = [\ u \quad v \quad w\ ] \qquad \omega_b = [\ p \quad q \quad r\ ]$$

Mentre la misurazione (una volta conosciuta la posizione del veicolo) dello spostamento lineare del quadricottero rispetto al sistema di riferimento inerziale è ottenibile direttamente, per ottenere lo spostamento angolare rispetto al sistema di riferimento inerziale è necessario fare riferimento a delle "matrici di rotazione".

2.2.2. Matrici di Rotazione

Queste particolari matrici quadrate descrivono, fissato un asse di rotazione, gli spostamenti angolari ottenibili effettuando rotazioni rispetto agli altri due assi che rimangono liberi. Gli angoli di Eulero descrivono la rotazione del sistema di riferimento solidale al corpo rigido rispetto a quello inerziale. Si otterranno quindi tre diverse matrici di rotazione :

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Una rotazione attorno a tutti e 3 gli assi è data dal prodotto delle 3 matrici appena indicate:

$$R(\phi,\theta,\psi) = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\theta & \sin\psi\sin\theta\cos\phi - \sin\phi\cos\psi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

si ottiene così la matrice R (detta anche DCM, o *Direction Cosine Matrix*), che permette di esprimere le componenti di un vettore nel sistema di riferimento inerziale a partire dalle sue componenti nel body frame, e viceversa. Possiamo pertanto scrivere:

$$\dot{\varepsilon} = R \cdot v_b$$

Nel modello dinamico del drone, la matrice R viene impiegata in particolare per esprimere il vettore spinta, che è definito nel body frame lungo l'asse z, nel sistema di riferimento inerziale, dove vengono scritte le equazioni della dinamica traslazionale.

Analogamente, per le trasformazioni angolari è presente un'altra matrice, che viene indicata con T (o ROT). La matrice T e la relativa matrice inversa sono definite come segue::

$$T = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \quad T^{-1} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}$$

La velocità angolare nel sistema inerziale viene indicata con ω, e possiamo scomporla per le tre rotazioni di base (roll, pitch e yaw) in modo da ottenere

$$\omega = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \dot{\phi} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \dot{\theta} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{\psi}$$

Adesso, con l'ausilio della matrice T (e della sua inversa) possiamo mettere in relazione i vettori di velocità angolare tra i sistemi di riferimento O_i e O_b grazie alle seguenti formule:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad \begin{bmatrix} p \\ q \\ r \end{bmatrix} = T^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.1}$$

Ricaviamo quindi che

$$\dot{\eta} = T \cdot \omega_b$$

2.2.3. Derivazione della Matrice T

Le tre rotazioni rispetto agli assi non vengono applicate contemporaneamente, ma si susseguono, ruotando in ordine rispetto ad un asse alla volta. Per convenzione noi applichiamo le rotazioni prima rispetto al'asse z, poi rispetto all'asse y, ed infine rispetto all'asse x.

L'idea di partenza è quella di prendere le componenti $\dot{\phi}$, $\dot{\theta}$ e $\dot{\psi}$ e di proiettarle nel body frame finale, utilizzando le matrici di rotazione presentate in precedenza. Siccome nel

body frame, in seguito all'ultima rotazione, l'asse attorno a cui avviene il roll coincide già con l'asse x del body frame finale, il termine $\dot{\phi}$ fornisce un contributo diretto:

$$\dot{\phi} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Per quanto riguarda l'angolo $\dot{\theta}$ rispetto all'asse y, siccome durante l'ultima rotazione il body frame è stato ruotato rispetto a x, per portarlo nel body frame finale, basta applicare la rotazione di roll mancante, cosa che facciamo grazie a $R_x(\phi)$:

$$\dot{\theta} R_x(\phi) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \dot{\theta} \begin{bmatrix} 0 \\ \cos\phi \\ -\sin\phi \end{bmatrix}$$

Infine, per quanto riguarda l'asse $\dot{\psi}$ rispetto all'asse z, per portarlo nel body frame finale, occorre applicare entrambe le rotazioni mancanti, ovvero $R_x(\phi) \cdot R_y(\theta)$.

$$\dot{\psi} R_x(\phi) R_y(\theta) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \dot{\psi} \begin{bmatrix} -\sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{bmatrix}$$

Riconducendoci dal sistema inerziale al body frame abbiamo ottenuto la matrice inversa T^{-1}. Sommando questi tre vettori ricavati (che ora sono tutti espressi nel body frame finale) si ottengono le componenti body della velocità angolare:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = T^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

invertendo si ricava la matrice T.

2.2.4. Roll, Pitch e Yaw

Come si può osservare nella Figura 2.3, il **roll** (o rollio) è il movimento che può far oscillare il quadricottero da destra a sinistra e viceversa, e si ottiene lasciando invariata la velocità del motore anteriore e di quello posteriore, e aumentando (o diminuendo) di una certa quantità la velocità del motore di destra, e diminuendo (o aumentando) la velocità del motore di sinistra in modo che la spinta verso l'alto non cambi.

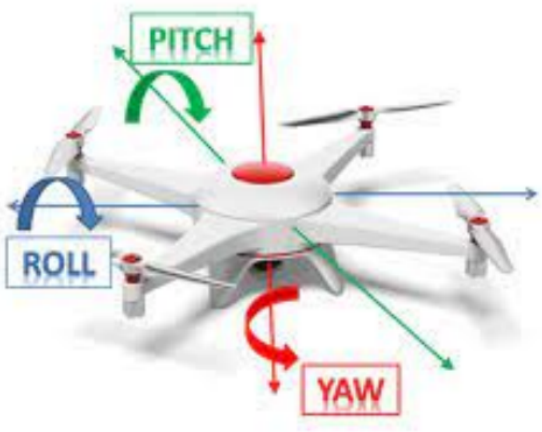


Figura 2.3. Versi di rotazione del quadricottero

Il **pitch** (o beccheggio) fa oscillare il quadricottero tra fronte e retro, in maniera analoga al roll. Infatti, per ottenere questo movimento l'idea è la stessa del rollio, ma si usano i motori anteriore e posteriore. Quindi si cambiano le velocità dei rotori anteriore e posteriore di una quantità tale da mantenere inalterata la spinta. Per angoli piccoli le due variazioni sono uguali. Infine, lo **yaw** (o imbardata) fa cambiare il movimento del drone rispetto all'asse z, mantenendo costante la spinta verso l'alto. Per ottenere questo movimento si aumenta la velocità dei rotori di due motori opposti, e si diminuisce la velocità dei due rotori rimanenti. In questo modo si avranno, per esempio, due rotori che girano velocemente in senso orario, e due che girano più lentamente in senso antiorario. Come risultato complessivo ci sarà dunque un momento che fa girare il drone intorno all'asse z.

Se le velocità sono opportunamente scelte, non ci sarà un cambiamento della spinta verso l'alto. La relazione tra la spinta e la velocità dei rotori ha un andamento quadratico, quindi per piccoli angoli desiderati, le variazioni di velocità dei rotori saranno circa uguali. Nella Figura 2.4 viene evidenziato meglio il comportamnto da far assumere ai diversi motori per ottenere la variazione di uno specifico angolo.

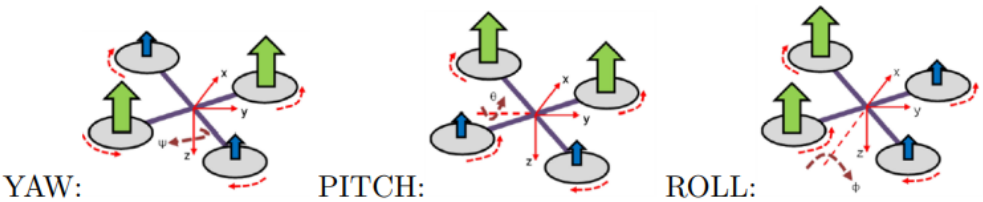


Figura 2.4. Variazione di roll, pitch e yaw

2.2.5. Spinta

L'ultimo movimento fondamentale che può essere controllato è la spinta verso l'alto (detta anche **thrust**). Come già accennato, la spinta è legata al quadrato della velocità angolare dei rotori, e dipende anche da una costante dell'aria e da come sono fatti i rotori. Per aumentare la spinta verso l'alto, si aumentano le velocità di tutti i rotori di una stessa quantità, in modo da non creare differenze che possano modificare l'assetto. Il concetto viene rappresentato in Figura 2.5.

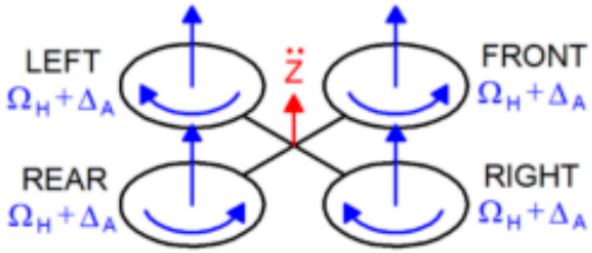


Figura 2.5. Variazione di thrust

2.2.6. Forze e Momenti Agenti sul Quadricottero

In questa sezione verranno ricavate le equazioni della dinamica del sistema, che consentiranno di ricavare in seguito il modello di controllo. Prima di procedere alla derivazione delle equazioni, occorre introdurre le seguenti ipotesi:

- Il centro di massa, il baricentro geometrico e l'intersezione dei 3 assi del sistema di riferimento solidale con il drone coincidono.
- Si ha perfetta simmetria rispetto ad ognuno dei 3 assi del sistema di riferimento solidale con il drone, che quindi comporta anche l'avere una matrice d'inerzia diagonale

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

- Il drone è un sistema tempo invariante, di cui quindi massa, dimensioni e struttura non sono soggette ad usura nel tempo. Inoltre la dinamica dei motori viene trascurata, siccome i loro transitori sono molo più rapidi di quelli del drone.

- Ogni singolo motore del drone è strutturalmente identico agli altri.

- Le condizioni del volo del velivolo sono nell'intorno dell'*hovering*, pertanto, le le velocità angolari del corpo p, q, r e gli angoli di roll ϕ e di pitch θ sono considerabili molto piccoli.

- Gli effetti giroscopici del corpo e delle eliche, che sono prodotti delle velocità angolari, sono anch'essi considerati trascurabili.

Effettuate queste ipotesi introduttive, possiamo definire la dinamica del sistema. La spinta totale del velivolo è data dalla somma delle singole spinte di ogni motore, che indichiamo con S_i. Per ipotesi dei motori quasi-statici, possiamo considerare la spinta di ciascun motore proporzionale alla sua velocità angolare al quadrato, pesata di un coefficiente di spinta, indicato con b:

$$S = \sum_{i=1}^{4} S_i = \sum_{i=1}^{4} b\omega_i^2 = b\sum_{i=1}^{4} \omega_i^2$$

Come analizzeremo in seguito nell'Equazione 2.4, dalla matrice di rotazione R è possibile ricavare l'accelerazione sull'asse z come:

$$m\ddot{z} = S\cos\phi\cos\theta - mg$$

Per ipotesi di angoli ϕ, θ piccoli, possiamo approssimare $S\cos\phi\cos\theta \approx S$. Quindi l'equazione appena ricavata diventa:

$$m\ddot{z} \approx S - mg$$

Considerando il sistema in condizione di hovering, si ottiene un'accelerazione nulla sull'asse z, pertanto la spinta verticale servirà esattamente a bilanciare la forza peso. Per quanto riguarda invece la dinamica rotazionale, considerando la formula generale di Eulero:

$$I\dot{\omega}_b + \omega_b \times I\omega_b = M + M_g$$

dove I è il momento di inerzia del sistema, M il momento totale, $\omega_b \times I\omega_b$ è l'effetto giroscopico del sistema e M_g il momento giroscopico, considerando l'ipotesi di effetti giroscopici nulli, l'equazione si riduce alla seguente formulazione:

$$I\dot{\omega}_b = M$$

Adesso, dall'ipotesi di perfetta simmetria si ha una matrice di inerzia diagonale, e l'equazione in forma matriciale diventa:

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}$$

Dall'Equazione 2.1 segue che:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

con l'ipotesi di angoli ϕ, θ piccoli, la matrice T è approssimabile con la matrice identità, pertanto si ricava $\dot{p} \approx \ddot{\phi}$, $\dot{q} \approx \ddot{\theta}$ e $\dot{r} \approx \ddot{\psi}$. Utilizzando quest'approssimazione, dall'equazione di Eulero della dinamica rotazionale del corpo rigido, si ricava che:

$$\ddot{\eta} \cdot I = \sum_{i=1}^{4} M_i = M$$

ovvero che il momento totale M agente su un sistema di corpi è dato dalla somma di tutti i momenti singoli agenti su ciascun corpo, ed equivale al prodotto tra il momento di inerzia totale e l'accelerazione angolare.

Analizzando gli effetti delle coppie di forze generate dai diversi motori, possiamo dedurre che i momenti di roll e pitch nascono dalla differenza di spinta tra due motori opposti, moltiplicata per la distanza l tra ciascun motore e il centro di massa. Ovvero:

$$M_\phi = lb(\omega_4^2 - \omega_2^2) \qquad M_\theta = lb(\omega_3^2 - \omega_1^2)$$

Il momento di yaw, invece, non è prodotto da una forza applicata ad un braccio, ma dalle coppie di reazione aerodinamica delle eliche, che ruotano a coppie in verso opposto. È quindi data da ciascun contributo pesato per il coefficiente di resistenza aerodinamica d:

$$M_\psi = d(\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2)$$

Questi momenti ricavati sono le componenti del vettore del momento totale M_b considerato nel sistema di riferimento solidale al telaio del drone.

$$M_b = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} lb(-\omega_2^2 + \omega_4^2) \\ lb(-\omega_1^2 + \omega_3^2) \\ \sum_{i=1}^{4} M_i \end{bmatrix}$$

2.3. Modello di Controllo

Adesso che abbiamo illustrato tutte le forze ed i momenti agenti sul quaadricottero, possiamo stabilire le 4 uscite di controllo U_i (chiamate **virtual input**) che i controllori dovranno generare per porre il drone nella posizione e nell'assetto desiderato:

$U_1 = b\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right)$	Il primo output di controllo è il **Thrust**, ovvero la spinta S che i motori devono generare per spostare il drone lungo l'asse z. È stato ricavato dalla seconda legge della dinamica.
$U_2 = lb\left(-\omega_2^2 + \omega_4^2\right)$	Il secondo output è dato dal momento che modifica l'inclinazione sull'asse x e provoca una variazione dell'angolo ϕ di **Roll**. È stato ricavato dalla legge del moto rotazionale.
$U_3 = lb\left(-\omega_1^2 + \omega_3^2\right)$	Il terzo output di controllo corrisponde al momento che modifica l'inclinazione sull'asse y e che provoca una variazione dell'angolo θ di **Pitch**. È stato ricavato dalla legge del moto rotazionale.
$U_4 = d\left(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2\right)$	Il quarto output è dato dal momento torcente totale, utilizzato per far ruotare il drone attorno all'asse z e che provoca una variazione dell'angolo ψ di **Yaw**. Anche questo è stato ricavato dalla legge del moto rotazionale.

Ponendo le uscite di controllo in forma matriciale otteniamo:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ -lb & 0 & lb & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}$$

Invertiamo ora la matrice in modo da ottenere i valori di rotazione dei singoli propulsori, poichè gli ingressi non possono essere inviati direttamente ai motori:

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4b} & 0 & -\frac{1}{2lb} & -\frac{1}{4d} \\ \frac{1}{4b} & -\frac{1}{2lb} & 0 & \frac{1}{4d} \\ \frac{1}{4b} & 0 & \frac{1}{2lb} & -\frac{1}{4d} \\ \frac{1}{4b} & \frac{1}{2lb} & 0 & \frac{1}{4d} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

Le relazioni ottenute costituiscono il modello matematico implementato all'interno dell'algoritmo di controllo [15].

2.3.1. Calcolo delle Costanti

Per concludere la sezione riguardante il modello matematico, adesso riportiamo i valori delle costanti utilizzate. Un accurato calcolo delle costanti è fondamentale per tarare il PID in maniera corretta. In particolare il coefficiente di spinta b può essere definito come:

$$b = \frac{S}{m \cdot v} = \frac{S}{\rho \cdot A \cdot v^2} = \frac{S}{\rho \cdot A \cdot (\omega_{rot} \cdot r)^2}$$

dove S è la spinta (o thrust) del drone, che abbiamo ricavato in precedenza; m è la massa del flusso d'aria che attraversa l'elica, esprimibile come il prodotto tra la densità dell'aria (ρ), la superficie (circolare) che si forma durante la rotazione (A) e la velocità del rotore (v), esprimibile a sua volta come prodotto tra la velocità di rotazione (ω_{rot}) e il raggio della pala (r).

Nota 2.2 Nel nostro caso il coefficiente b ha un valore pari a 0.00001163, poichè

- S corrisponde al 50% del thrust complessivo, in quanto per fare i test è stata considerata una fase di hovering ideale ed è stata effettuata l'ipotesi che i motori lavorino a metà della loro potenza effettiva.
- $r^2 = 0.127\ m^2$
- $\omega_{rot}^2 = 36606\ rad^2/s^2$
- $\rho = 1.225\ \text{kg}/m^3$

Per quanto riguarda il coefficiente di resistenza aerodinamica d, possiamo invece definirlo come:

$$d = \frac{1}{2} \cdot C_d \cdot \rho \cdot A \cdot v^2$$

Con C_d che rappresenta il coefficiente di resistenza aerodinamica e vale $1.19 \cdot 10^{-10}$. Si ha pertanto che d vale 0.00008. Tutte le costanti riportate in questa sezione relative al drone sono state ricavate dalla relazione [11] in bibliografia.

2.4. Modello Matematico della Traiettoria

Ora che abbiamo introdotto la dinamica di volo del drone e quella di controllo, introduciamo il modello matematico per l'inseguimento della traiettoria del drone in catena aperta. L'obbiettivo è quello di simulare lo spostamento del drone di una distanza pari a D (indicata in metri) sugli assi x e y in un periodo di tempo indicato con T. Partendo da queste supposizioni, si è deciso di impiegare un modello matematico che converta un preciso spostamento in metri sugli assi x e y in traiettorie da far seguire al drone sugli angoli di roll (rotazione sull'asse x, che comporta uno spostamento sull'asse y) e pitch (rotazione sull'asse y, che comporta uno spostamento sull'asse x) . Procediamo per punti.

2.4.1. Relazioni tra Angoli e Accelerazioni

Innanzitutto, vogliamo ottenere una relazione tra gli angoli ϕ e θ e le accelerazioni del sistema sugli assi x e y. Abbiamo ricavato dalla seconda legge della dinamica che la massa del sistema moltiplicata per la sua accelerazione è pari alla somma di tutte le forze agenti su esso:

$$m \cdot a = m \cdot \ddot{p} = m \cdot R(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ S \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

in questo caso abbiamo considerato anche la forza peso agente sul drone. Per l'inseguimento della traiettoria solo sugli assi x e y, si è deciso di impostare la matrice R come

$$R(\phi, \theta, \psi) = R_x(\phi) R_y(\theta)$$

in modo da ottenere così (separando l'accelerazione complessiva nelle tre componenti a_x, a_y e a_z):

$$a_x = \frac{S}{m} \cos\phi \sin\theta \tag{2.2}$$

$$a_y = \frac{S}{m} \sin\phi \tag{2.3}$$

$$a_z = \frac{S}{m} \cos\phi \cos\theta - g \tag{2.4}$$

Siccome durante le prove il drone non si è trovato in esperienza di volo libero, ma è rimasto attaccato ad un supporto, bisogna introdurre l'ipotesi di quota costante ($a_z = 0$). Pertanto, dalla relazione 2.4 otteniamo che:

$$\frac{S}{m} = \frac{g}{\cos\phi \cos\theta} \tag{2.5}$$

Adesso, per ottenere la relazione tra l'accelerazione lungo l'asse x e l'angolo θ, sostituendo la relazione appena ricavata nella 2.2 otteniamo che:

$$a_x = \frac{\cancel{\cos\phi}\sin\theta}{\cancel{\cos\phi}\cos\theta} \cdot g \quad \rightarrow \quad \frac{a_x}{g} = \tan\theta$$

Ora, se impostiamo l'equazione 2.5 in modo da isolare g otteniamo:

$$g = \frac{S}{m}\cos\phi\cos\theta \tag{2.6}$$

Adesso, per ricavare il rapporto tra a_y e l'angolo ϕ procediamo elevando al quadrato le tre relazioni 2.2, 2.3 e 2.6, e sommandole tra di loro:

$$a_x^2 + a_y^2 + g^2 = \left(\frac{S}{m}\right)^2 \cdot (\cos^2\phi\sin^2\theta + \sin^2\phi + \cos^2\phi\cos^2\theta)$$

Raccogliendo il termine $\cos^2\phi$ otteniamo $\sin^2\theta + \cos^2\theta = 1$, grazie al quale l'ultimo termine nell'equazione diventa $\cos^2\phi + \sin^2\phi = 1$, e pertanto, abbiamo ricavato che:

$$a_x^2 + a_y^2 + g^2 = \left(\frac{S}{m}\right)^2 \quad \rightarrow \quad \frac{S}{m} = \sqrt{a_x^2 + a_y^2 + g^2}$$

Adesso, andando a sostituire nella relazione 2.3, riusciamo a ricavare un'espressione dell'angolo ϕ dipendente unicamente dall'accelerazione sull'asse y:

$$\sin\phi = \frac{a_y}{S/m} \quad \rightarrow \quad \phi = \arcsin\frac{a_y}{\sqrt{a_x^2 + a_y^2 + g^2}}$$

Abbiamo quindi ricavato due relazioni che legano le accelerazioni sugli assi x e y agli angoli di pitch e di roll.

2.4.2. Traiettoria a Strappo Minimo

Adesso che abbiamo ricavato un'espressione degli angoli da assumere in funzione delle accelerazioni richieste, possiamo introdurre il modello della traiettoria. Nel caso di questo progetto si è scelta di utilizzare la **traiettoria a strappo minimo** [10, 16, 19]. In fisica, lo strappo (o jerk) è la derivata dell'accelerazione, e indica quanto l'accelerazione sta aumentando o diminuendo. Siccome abbiamo ricavato che l'angolo da assumere è proporzionale all'accelerazione, allora la velocità con cui il drone deve ruotare su quest'angolo sarà proporzionale allo strappo.

L'obbiettivo della traiettoria a strappo minimo è quello di far effettuare al veicolo rotazioni il più possibile lente e dolci, in modo che queste risultino facili da inseguire per il PID di

assetto. Quindi, considerata una traiettoria $x(t)$, vogliamo che questa sia il meno brusca possibile, ovvero, vogliamo minimizzare un funzionale obbiettivo

$$\min \int_0^T x^{(3)}(t)^2 \, dt$$

Un funzionale si differenzia da una funzione per il fatto che quest'ultima prende come ingresso un solo parametro, mentre un funzionale prende come ingresso un'intera traiettoria (nel nostro caso l'ingresso è dato dal jerk della traiettoria $x(t)$). Prima di procedere con la soluzione, dobbiamo imporre sei condizioni al contorno. Siccome vogliamo che il drone parta da fermo in una posizione identificabile come l'origine (quindi zero) e arrivi in un istante di tempo T alla posizione D da fermo e con accelerazione nulla, imponiamo le seguenti condizioni:

$$x(0) = \dot{x}(0) = \ddot{x}(0) = 0 \qquad x(T) = D,\ \dot{x}(T) = \ddot{x}(T) = 0$$

Considerato quindi il funzionale

$$J[x] = \int_0^T x^{(3)}(t)^2 \, dt$$

noi vogliamo trovare una soluzione che minimizzi il valore di $J[x]$, ovvero il valore che rappresenta quanto è brusca una traiettoria nel suo complesso.

Nota 2.3 Per risolvere questo problema applichiamo l'equazione di Eulero-Lagrange, che per un integrando L che dipende da x e dalle sue prime tre derivate è rappresentabile come:

$$\frac{\partial L}{\partial x} - \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} + \frac{d^2}{dt^2}\frac{\partial L}{\partial \ddot{x}} - \frac{d^3}{dt^3}\frac{\partial L}{\partial x^{(3)}} = 0$$

Per risolvere il problema per prima cosa perturbiamo la soluzione ottima, che indichiamo con $x^*(t)$. In questo modo la traiettoria diventa:

$$x(t) = x^*(t) + \epsilon \cdot \delta(t)$$

dove ϵ è un valore arbitrariamente piccolo e $\delta(t)$ indica una deformazione qualsiasi. Adesso, se x^* rappresenta la soluzione ottima, e quindi minima, vuol dire che deve avere derivata nulla rispetto a ϵ. Siccome il costo della curva deformata è una funzione di ϵ:

$$J(\epsilon) = \int_0^T (x^{*(3)} + \epsilon \cdot \delta^{(3)})^2 \, dt$$

poichè x^* è il minimo, $J(\epsilon)$ deve avere il minimo in $\epsilon = 0$. Quindi per avere il costo minimo, dobbiamo porre la derivata di $J(\epsilon)$ su ϵ nulla in $\epsilon = 0$, ovvero:

$$\left.\frac{dJ}{d\epsilon}\right|_{\epsilon=0} = 2\int_0^T x^{*(3)} \cdot \delta^{(3)}\ dt = 0$$

Integrando per parti tre volte si ottiene che:

$$\int x^{*(3)}\ \delta^{(3)}\ dt = -\int x^{*(4)}\ \ddot{\delta}\ dt = +\int x^{*(5)}\ \dot{\delta}\ dt = -\int x^{*(6)}\ \delta\ dt$$

notiamo che ad ogni passaggio compare un termine di bordo, che scompare perchè δ, $\dot{\delta}$ e $\ddot{\delta}$ sono nulli agli estremi. Quindi per soddisfare la condizione deve valere

$$\int_0^T x^{*(6)}(t)\ \delta(t)\ dt = 0 \qquad\qquad \forall \delta$$

Pertanto, per far valere la condizione appena ricavata per ogni deformazione δ, occorre necessariamente che valga $x^{*(6)} = 0$, ovvero che

$$\frac{d^6x}{dt^6} = 0$$

Se la derivata sesta è nulla, vuol dire che $x(t)$ è un polinomio di quinto grado con sei costanti

$$x(t) = c_0 + c_1t + c_2t^2 + c_3t^3 + c_4t^4 + c_5t^5$$

avendo sei condizioni al contorno, otteniamo che la soluzione al problema è unica.

2.4.3. Applicazione del Modello al Sistema

Adesso, per semplificare i calcoli, utilizziamo una variabile temporale $\tau = t/T$, in modo che questa risulti normalizzata tra 0 e 1 durante tutto l'intervallo T dello spostamento. Prendendo come esempio il movimento sull'asse x (ma le considerazioni sono analoghe per l'asse y), otteniamo quindi che lo spostamento è dato da $x = D \cdot s(\tau)$, dove:

$$s(\tau) = c_0 + c_1t + c_2t^2 + c_3t^3 + c_4t^4 + c_5t^5$$

Derivando rispetto al tempo si ha:

$$x = D \cdot s \qquad \dot{x} = \frac{D}{T}\dot{s} \qquad \ddot{x} = \frac{D}{T^2}\ddot{s} \qquad x^{(3)} = \frac{D}{T^3}s^{(3)}$$

Adesso, per ricavare i coefficienti della funzione $s(\tau)$, la deriviamo due volte e applichiamo le condizioni al contorno:

$$\begin{aligned} s(\tau) &= c_0 + c_1\tau + c_2\tau^2 + c_3\tau^3 + c_4\tau^4 + c_5\tau^5 \\ s'(\tau) &= c_1 + 2c_2\tau + 3c_3\tau^2 + 4c_4\tau^3 + 5c_5\tau^4 \\ s''(\tau) &= 2c_2 + 6c_3\tau + 12c_4\tau^2 + 20c_5\tau^3 \end{aligned}$$

Nota 2.4 Imponendo le condizioni iniziali $s(0) = s'(0) = s''(0) = 0$, si ricava che $c_0 = c_1 = c_2 = 0$. Inoltre, imponendo anche le condizioni finali $s(1) = 1$, $s'(1) = 0$, $s''(1) = 0$ si ottiene il seguente sistema:

$$\begin{cases} c_3 + c_4 + c_5 = 1 \\ 3c_3 + 4c_4 + 5c_5 = 0 \\ 6c_3 + 12c_4 + 20c_5 = 0 \end{cases}$$

dal quale ricaviamo che $c_3 = 10,\ c_4 = -15$ e $c_5 = 6$. La funzione normalizzata della posizione risulta quindi:

$$s(\tau) = 10\tau^3 - 15\tau^4 + 6\tau^5$$

Derivandola due volte rispetto a τ otteniamo:

$$s''(\tau) = 60\tau - 180\tau^2 + 120\tau^3$$

Dalla relazione $\ddot{x} = \frac{D}{T^2}\, s''(\tau)$, ricaviamo che

$$\ddot{x}(t) = \frac{D}{T^2}\left(60\tau - 180\tau^2 + 120\tau^3\right)$$

e quindi, siccome abbiamo ricavato in precedenza che l'andamento dell'angolo di pitch è dato dall'arcotangente del rapporto tra $\ddot{x}$ e g, otteniamo che:

$$\theta(t) = \arctan\left(\frac{D\left(60\tau - 180\tau^2 + 120\tau^3\right)}{g\,T^2}\right)$$

Analogamente si ricava l'andamento dell'angolo di roll per lo spostamento sull'asse y.

2.5. Modello del Controllo a Ciclo Chiuso

Come accennato nel paragrafo precedente, la dinamica del sistema implica una corrispondenza angolo-accelerazione, quindi per passare dalla posizione richiesta all'angolo desiderato è richiesta una doppia derivazione, operazione che risulta semplice su una traiettoria analitica nota, mentre invece per effettuare il passaggio inverso è richiesta una doppia integrazione, e quindi ogni errore sull'angolo viene integrato due volte e si accumula sulla posizione.

Per il controllo in catena aperta è stato utilizzato il modello di traiettoria a jerk minimo, una soluzione che ottimizza la traiettoria da inseguire, ma con un limite strutturale evi-

dente. Lavorando a ciclo aperto, il massimo grado di azione applicabile al sistema era assumere l'accelerazione comandata pari all'accelerazione di riferimento, quindi un errore di accelerazione nullo.

$$\ddot{e} = \ddot{x}_{ref} - a_x = 0 \tag{2.7}$$

Tuttavia, non essendoci retroazione, non c'era un modo diretto per correggere gli errori di posizione o di velocità man mano che si presentavano durante il moto.

Per garantire un'azione di controllo sull'errore, serve quindi introdurre un termine correttivo all'equazione 2.7, e siccome da un sistema a doppio integratore possiamo agire solo sull'errore e e sulla sua velocità $\dot{e}$, allora il modo più semplice è introdurre nell'equazione una loro combinazione lineare, pesata da coefficienti K_d e K_p. Con questa aggiunta, l'equazione precedente diventa:

$$\ddot{e} + K_d\dot{e} + K_p e = 0$$

Questa equazione della dinamica del secondo ordine ha lo stesso comportamento di un sistema massa-molla-smorzatore, cosa vantaggiosa in quanto (sotto la condizione che $K_p,\ K_d > 0$) l'errore converge sempre a 0, indipendentemente dalla traiettoria seguita.

Siccome la scelta del modello di controllo è ricaduta su un sistema del secondo ordine, è possibile ricondurci alla scelta dei parametri tramite la pulsazione naturale ω_n e lo smorzamento ζ, così che, imponendo

$$K_p = \omega_n^2 \qquad K_d = 2\zeta\omega_n$$

e sostituendo nell'equazione della dinamica del secondo ordine, ricaviamo:

$$\ddot{e} + 2\zeta\omega_n\dot{e} + \omega_n^2 e = 0$$

così da ottenere come radici del polinomio caratteristico associato:

$$s_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}$$

e ricondursi esattamente al caso di un sistema del secondo ordine (oscillatore armonico smorzato), nel quale la presenza di oscillazioni e la rapidità di convergenza della risposta sono determinate unicamente dai parametri ζ e ω_n. Grazie a questa parametrizzazione, la forma della risposta dell'errore (determinata unicamente dal coefficiente di smorzamento ζ) e la sua rapidità (determinata dalla pulsazione naturale ω_n) risltano perfettamente disaccoppiate [15].

3

Strumentazione Hardware

In questo capitolo vengono descritti i principali componenti hardware del quadricottero e il loro ruolo all'interno del sistema. L'architettura può essere suddivisa in due catene. La catena di potenza parte dalla batteria, la cui energia viene distribuita dalla PDB ai quattro ESC, che a loro volta alimentano i motori brushless e le eliche. La catena di controllo, invece, è gestita dalla scheda STM32, che acquisisce l'assetto dall'IMU, comunica con MATLAB tramite collegamento seriale, riceve i segnali di emergenza dal ricevitore del radiocomando e genera i segnali PWM di comando per gli ESC [14].

3.1. Scheda STM32 H745 Nucleo-144

La scheda STM32 H745 Nucleo-144, prodotta da STMicroelectronics, costituisce l'unità di elaborazione del sistema. Su di essa vengono eseguiti la stima dell'assetto e della posizione, i regolatori di controllo, la macchina a stati e la gestione della comunicazione con MATLAB. Il nome Nucleo-144 deriva dal package LQFP a 144 pin del microcontrollore montato, lo STM32H745ZIT6. Si tratta di un microcontrollore dual-core che integra un ARM Cortex-M7 con frequenza di clock fino a 480 MHz e un ARM Cortex-M4 fino a 240 MHz [6].

Ai fini del progetto, le risorse rilevanti della scheda sono le periferiche che la interfacciano con il resto del sistema: i timer, utilizzati sia per generare i segnali PWM diretti agli ESC sia per acquisire i segnali provenienti dal ricevitore; l'interfaccia I2C, utilizzata per la comunicazione con l'IMU; la periferica UART, accessibile tramite il collegamento USB, utilizzata per lo scambio di dati con MATLAB. La memoria disponibile consente inoltre di allocare a bordo l'intera traiettoria di riferimento prima dell'esecuzione della manovra.

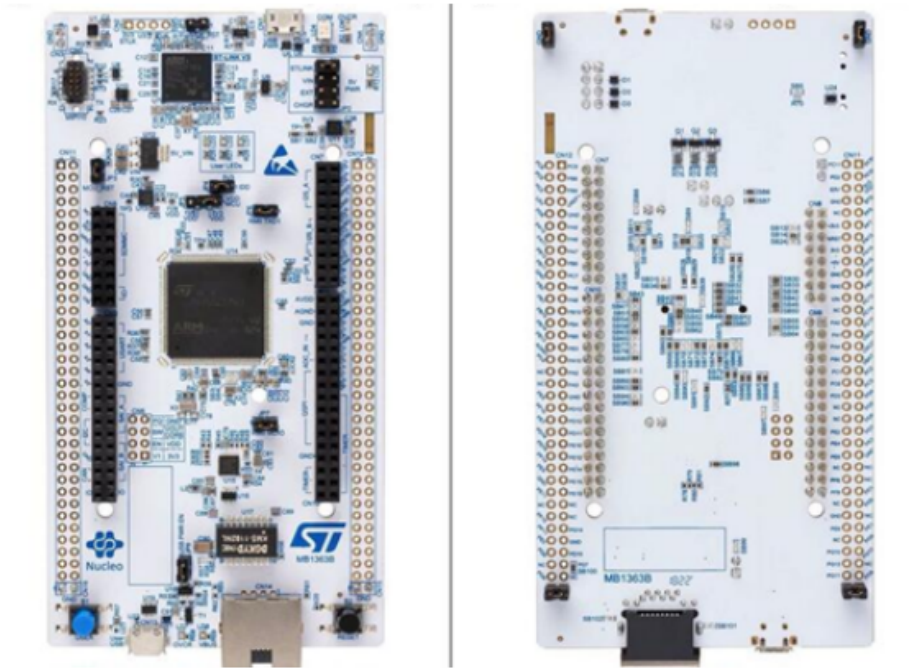


Figura 3.1. Fronte e retro della scheda STM32 H745 Nucleo-144.

3.2. Motori

Il quadricottero è equipaggiato con quattro motori brushless Turnigy D3536/9 910KV, appartenenti alla classe 35 (diametro di 35 mm e lunghezza di 36 mm).

Un motore brushless è composto da uno statore, che porta gli avvolgimenti di rame, e da un rotore a magneti permanenti. A differenza dei motori a spazzole, la commutazione della corrente tra le fasi non avviene meccanicamente, ma elettronicamente, ovvero, gli avvolgimenti vengono alimentati in sequenza, generando un campo magnetico rotante che trascina il rotore. La sequenza e la frequenza di commutazione, sincronizzate con la posizione del rotore, determinano verso e velocità di rotazione. Questa funzione è svolta

dall'ESC, descritto nella sezione successiva. L'assenza di contatti striscianti garantisce elevata efficienza, maggiore durata e ridotta manutenzione.



Figura 3.2. Motore brushless Turnigy D3536/9 910KV e relativa struttura interna.

I limiti operativi del motore, ovvero corrente massima, tensione massima e resistenza interna, sono riportati nella scheda tecnica di Figura 3.3 e vincolano la scelta di ESC e batteria.

RPM	910kv
Fasi	3
Massima corrente	25.5A
Potenza massima	370W a 15V
Peso (compreso connettori)	102g
Batteria	2-4 Cell /7.4 14.8V
Diametro del pozzo	5mm
No corrente di carico	1.5A
Dimensioni	35x36mm
Spinta max	1050g
Dimensione prop.	7.4V/12x5 14.8V/10x7
Resistenza interna	0,063 Ohm

Figura 3.3. Scheda tecnica del motore Turnigy D3536/9 910KV.

Nota 3.1 La costante $K_V = 910$ rpm/V indica la velocità di rotazione a vuoto per ogni volt di tensione applicata, quindi, alimentandolo con la tensione nominale della batteria di 14.8 V, il motore raggiunge a vuoto circa $910 \cdot 14.8 \simeq 13\,500$ rpm.

3.3. ESC

L'ESC (*Electronic Speed Controller*) è il dispositivo che pilota il motore brushless. Questo componente riceve in ingresso un segnale dalla scheda di controllo e, sulla base di questo comando, commuta la corrente prelevata dalla batteria tra le tre fasi del motore, regolandone la velocità di rotazione. Ciascun motore è pilotato da un ESC Turnigy Plush-30A, in grado di erogare una corrente continua di 30 A e compatibile con batterie LiPo da 2 a 4 celle, quindi con la batteria 4S utilizzata.

L'ESC integra inoltre un BEC (*Battery Eliminator Circuit*), ovvero un regolatore che fornisce in uscita una tensione ausiliaria stabilizzata, ricavata dalla batteria principale. Nel progetto questa tensione viene utilizzata per alimentare la scheda STM32, come verrà descritto nella sezione 3.9.



Figura 3.4. ESC Turnigy Plush-30A.

Amperaggio	30 A
Tensione	8.4 – 16.8 V
BEC	5.5 V /4.0 A
Tensione di Cutoff	3.2 V
Peso	24.5g
Dimensioni	36x23.5x10mm
Frequenza	48 MHz

Figura 3.5. Scheda tecnica dell'ESC Turnigy Plush-30A.

L'ESC è programmabile e consente di configurare diversi parametri, tra cui il tipo di batteria, la curva di accelerazione, la frenatura, la modalità di avvio e il limite di corrente, così da adattarne il comportamento alle esigenze richieste. La programmazione, effettuata tramite l'apposita scheda fornita dal produttore, è stata ereditata dai progetti precedenti. Tutti gli ESC sono stati configurati secondo le impostazioni riportate in Figura 3.6.

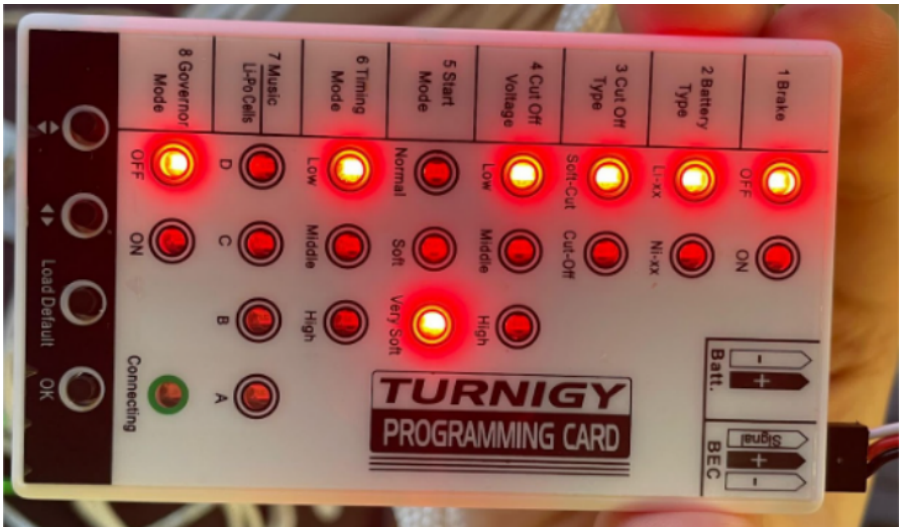


Figura 3.6. Scheda di programmazione degli ESC e configurazione adottata.

3.4. Batteria

L'alimentazione del sistema viene fornita da una batteria LiPo FullPower 4S 14.8V 3300mAh 35C Silver Edition V2 con connettore Deans. La sigla 4S indica che la batteria è costituita da quattro celle collegate in serie, ciascuna con una tensione nominale complessiva di 14.8 V, corrispondente a 3.7 V per cella. La capacità di 3300 mAh indica la carica erogabile, mentre il coefficiente di scarica pari a 35C indica che la batteria può fornire una corrente continua massima pari a 35 volte la sua capacità nominale, in questo caso pari a $35 \cdot 3.3$ A $\simeq$ 115 A. La corrente massima disponibile è sufficiente a coprire l'assorbimento massimo teorico dei quattro ESC, pari a $4 \cdot 30, \mathrm{A} = 120, \mathrm{A}$, valore che non viene raggiunto durante il funzionamento a causa del limitato intervallo di duty cycle utilizzato. La batteria risulta quindi adeguata alle esigenze del sistema.



Figura 3.7. Batteria FullPower 4S 14.8V 3300mAh 35C Silver Edition V2 Deans.

Capacità	$3300mAh$
Tensione	$4celle/14.8V$
Peso	$360g$
Dimensioni	$43x138x28mm$

Figura 3.8. Scheda tecnica della batteria.

3.5. Eliche

Le eliche convertono la coppia fornita dai motori in spinta. In questo progetto sono state impiegate le eliche Slow Fly Electric Prop 9057SF, che hanno un diametro di 9 pollici e passo di 5.7 pollici, dove il passo rappresenta l'avanzamento teorico dell'elica in un giro completo, e la sigla SF (*Slow Fly*) identifica eliche progettate per regimi di rotazione contenuti.

Come illustrato in Figura 3.9, le quattro eliche sono disposte in coppie controrotanti, ovvero, i motori situati lungo una diagonale ruotano in senso orario, mentre quelli sulla diagonale opposta girano in senso antiorario. Questa disposizione bilancia le coppie di reazione aerodinamica generate dalle eliche, che altrimenti indurrebbero una rotazione indesiderata del telaio attorno all'asse verticale. Lo sbilanciamento volontario delle velocità tra le due coppie è proprio il meccanismo con cui viene controllato l'angolo di yaw, mentre la differenza di spinta tra motori opposti genera i momenti di roll e pitch.

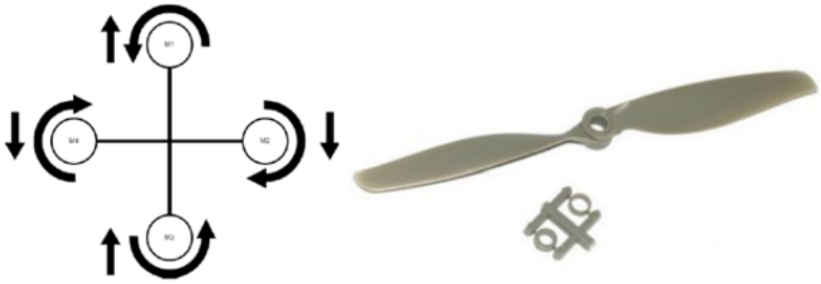


Figura 3.9. Schema del senso di rotazione delle eliche ed elica Slow Fly Electric Prop 9057SF.

3.6. IMU

L'IMU (*Inertial Measurement Unit*) è il sensore che fornisce al sistema di controllo la misura dell'assetto. Un'IMU combina un accelerometro triassiale, che misura le accelerazioni lineari (inclusa la componente gravitazionale), e un giroscopio triassiale, che misura le velocità angolari attorno ai tre assi. Integrando le velocità angolari e utilizzando la direzione della gravità come riferimento, è possibile stimare gli angoli di roll, pitch e yaw.

Nel progetto è stata impiegata una scheda Adafruit basata sul chip Bosch BNO055, un sensore a 9 gradi di libertà che integra, oltre ad accelerometro e giroscopio, un magnetometro triassiale e un microcontrollore ARM Cortex-M0+ dedicato. Su quest'ultimo è eseguito il software di sensor fusion Bosch Sensortec, che combina le misure dei tre sensori e restituisce direttamente l'orientamento del sensore sotto forma di angoli di Eulero o quaternioni. La stima dell'assetto è quindi delegata all'IMU, e la scheda STM32 si limita a leggere gli angoli tramite interfaccia I2C. La scheda Adafruit include inoltre il regolatore di tensione a 3.3 V, i traslatori di livello per le linee I2C e di reset e un oscillatore al quarzo esterno a 32.768 kHz, che migliora la precisione temporale del sensore.

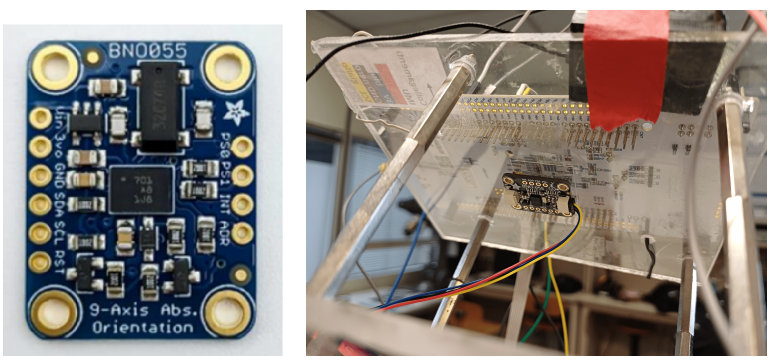


Figura 3.10. IMU Adafruit BNO055 e relativa installazione sul quadricottero.

3.7. PDB

Il PDB (*Power Distribution Board*) è la scheda che distribuisce la tensione della batteria tra i quattro ESC. Tramite piste conduttive di dimensione adeguata alle correnti in richieste, collega in parallelo i quattro ESC alla batteria, garantendo che tutti ricevano la stessa tensione di alimentazione e semplificando il cablaggio del telaio.

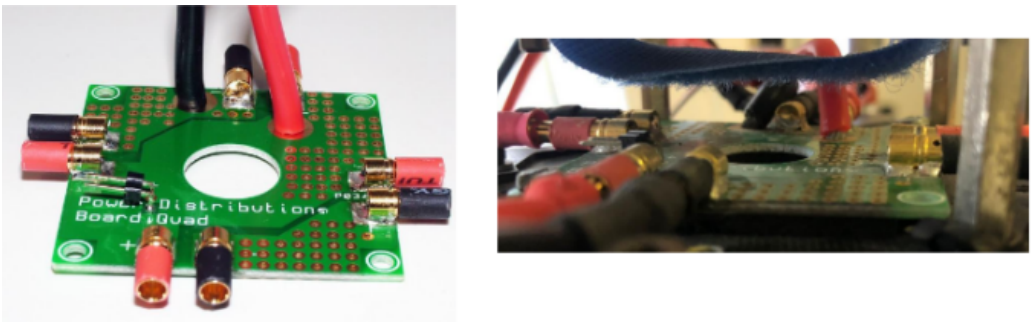


Figura 3.11. PDB e relativa installazione sul quadricottero.

3.8. Radiocomando

Il radiocomando è composto da un trasmettitore (Tx), azionato dall'operatore, e da un ricevitore (Rx), montato sul quadricottero. Il trasmettitore codifica la posizione di leve e interruttori e la invia via radio; il ricevitore la decodifica e la rende disponibile su uscite separate, una per canale, ciascuna sotto forma di segnale PWM il cui duty cycle è proporzionale alla posizione del comando corrispondente. Questi segnali vengono acquisiti dalla scheda STM32, che ne misura il duty cycle e lo interpreta come comando.

Nelle prime fasi del progetto il radiocomando forniva sia i comandi di armo e spegnimento dei motori, tramite l'interruttore associato al canale 5, sia i riferimenti angolari. Con l'introduzione della comunicazione con MATLAB, il suo ruolo è diventato quello di spegnimento di emergenza, siccome consente di forzare l'armamento o lo spegnimento dei motori in qualunque momento, con priorità sui comandi provenienti da MATLAB e senza che l'operatore debba avvicinarsi al drone durante il funzionamento.

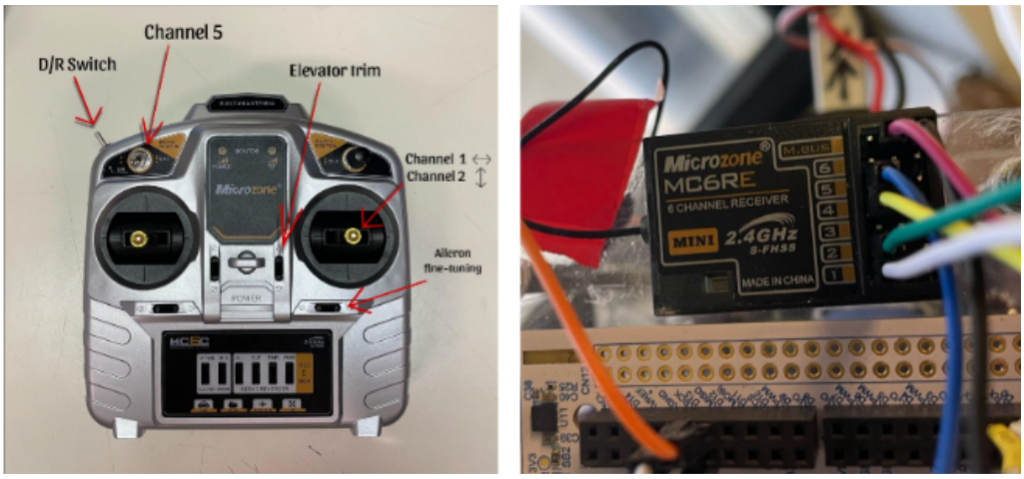


Figura 3.12. Trasmettitore e ricevitore del radiocomando.

3.9. Schema dei Collegamenti

Lo schema dei collegamenti è riportato in Figura 3.13. Rispetto alla configurazione descritta nella relazione dell'anno precedente [11], sono stati invertiti i pin di collegamento dei canali 4 e 5 del ricevitore: in precedenza il canale 4 era collegato al pin PA0 e il canale 5 al pin PA3, mentre ora il canale 4 è collegato a PA3 e il canale 5 a PA0. I restanti collegamenti sono invariati. La batteria, inoltre, non è più fissata al telaio, ma al supporto verticale del banco di prova tramite nastro biadesivo, come mostrato in Figura 3.14.

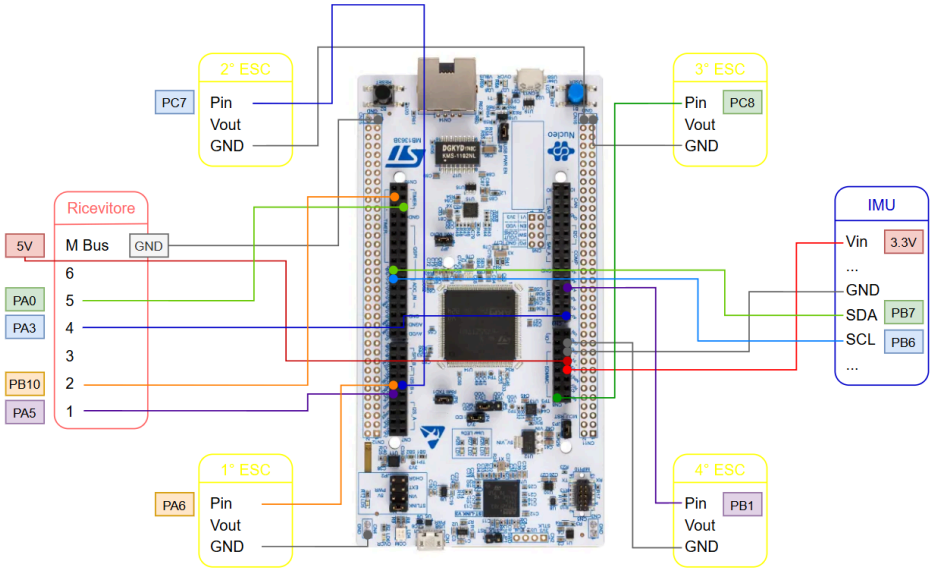


Figura 3.13. Schema dei collegamenti.

La scheda STM32 può essere alimentata tramite il connettore USB oppure collegando con un jumper il pin Vin (situato sopra il collegamento di massa del quarto ESC) all'uscita Vout del BEC di uno degli ESC.

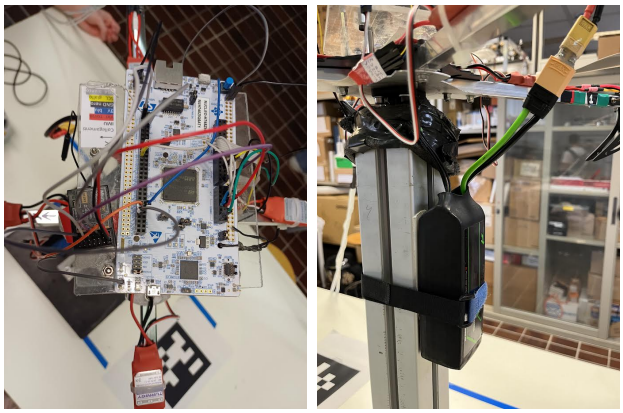


Figura 3.14. Collegamenti del drone e sistemazione della batteria.

3.10. Prova dei Motori

Ciascun motore è comandato da un segnale PWM, generato da un timer della STM32 e inviato al rispettivo ESC. Il PWM è un'onda rettangolare di periodo T, che rimane a livello alto per un tempo T_{ON}; il rapporto $\delta = T_{ON}/T$ è detto **duty cycle**. Il segnale non serve ad alimentare direttamente il motore, ma permette all'ESC di interpretare la durata dell'impulso e di determinare il livello di potenza richiesto, prelevando dalla batteria l'energia necessaria. Variando il duty cycle si regolano quindi la velocità del motore e la spinta dell'elica.

Nota 3.2 L'ESC richiede un segnale a 50 Hz, corrispondente a un periodo di 20 ms. Il comando viene definito tramite impulsi di durata compresa tra circa 1 ms e 2 ms, che corrispondono a valori di duty cycle compresi tra il 5% e il 10%. Nella scheda il duty cycle è espresso in percentuale e convertito nel valore di confronto da scrivere nel registro $CCRx$ del timer secondo la relazione

$$CCRx = \delta_{\%} \cdot \frac{ARR}{100},$$

dove ARR (*Auto-Reload Register*) è il registro che definisce il periodo di conteggio del timer.

Prima dell'utilizzo, l'ESC deve essere calibrato per riconoscere gli estremi dell'intervallo di duty cycle utilizzato. Nel presente lavoro è stata adottata la procedura descritta nella relazione di un progetto precedente [11], riportata di seguito.

3.10.1. Calibrazione ESC

La calibrazione va eseguita solo al primo utilizzo dell'ESC, così che quest'ultimo possa memorizzare gli estremi del range di comando. La procedura è divisa nei seguenti passaggi:

1. impostare il PWM al duty cycle massimo desiderato;
2. alimentare l'ESC solo dopo aver impostato il valore massimo. Affinchè l'ESC entri in modalità di calibrazione, il duty cycle deve essere almeno pari all'8%;
3. l'ESC emette il suono di accensione seguito da un doppio beep;
4. attendere almeno 2 s (nel codice sono stati utilizzati 3 s per sicurezza);
5. impostare il PWM al duty cycle minimo desiderato;
6. l'ESC emette tre beep seguiti da un beep prolungato, a conferma dell'avvenuta calibrazione, e resta in attesa di un segnale di comando. In assenza di quest'ultimo, emette un beep ogni 2 s.

La procedura è implementata dalla funzione `ESC_Calibrate`, contenuta nel file `ESC.c` insieme alle altre funzioni di gestione degli ESC:

```
void ESC_Calibrate (){
    TIM3 -> CCR1 = ( uint32_t ) (TIM3 -> ARR * LIMIT_DUTY / 100);
    TIM3 -> CCR2 = ( uint32_t ) (TIM3 -> ARR * LIMIT_DUTY / 100);
    TIM3 -> CCR3 = ( uint32_t ) (TIM3 -> ARR * LIMIT_DUTY / 100);
    TIM3 -> CCR4 = ( uint32_t ) (TIM3 -> ARR * LIMIT_DUTY / 100);
    HAL_Delay (3000);
    TIM3 -> CCR1 = ( uint32_t ) (TIM3 -> ARR * OFF_DUTY / 100);
    TIM3 -> CCR2 = ( uint32_t ) (TIM3 -> ARR * OFF_DUTY / 100);
    TIM3 -> CCR3 = ( uint32_t ) (TIM3 -> ARR * OFF_DUTY / 100);
    TIM3 -> CCR4 = ( uint32_t ) (TIM3 -> ARR * OFF_DUTY / 100);
}
```

I quattro canali del timer TIM3 vengono portati al duty cycle massimo, mantenuti per 3 s e infine riportati al minimo. I valori di duty cycle adottati, ereditati dai progetti precedenti [relazione1], e sono riassunti in Tabella 3.1. Gli estremi della calibrazione sono definiti nel file `ESC.h`:

```
# define OFF_DUTY 5
# define LIMIT_DUTY 10
```

Parametro	**Duty cycle**
Limite inferiore di calibrazione	5%
Limite superiore di calibrazione	10%
Valore di armo	5.3%
Valore minimo di controllo	5.4%
Valore massimo di controllo	6.5%

Tabella 3.1. Valori di duty cycle utilizzati per gli ESC.

Si osserva che l'intervallo effettivamente utilizzato dai regolatori (5.4–6.5%) rappresenta solo una parte limitata del range calibrato (5–10%). Questo limite, necessario per garantire la sicurezza durante le prove al banco, riduce però la spinta massima disponibile e può causare dei fenomeni di saturazione, che verranno descritti nei capitoli successivi, in particolare sull'asse di yaw. La calibrazione all'avvio viene gestita nel file `main.c` tramite la variabile `calibrate`.

4

Architettura Implementativa

Nel seguente capitolo vengono descritte le modifiche e le integrazioni apportate al codice di sistema per raggiungere gli obiettivi di controllo previsti dal progetto. Gli interventi sono presentati in ordine cronologico, così da ricostruire l'evoluzione dell'architettura dalla gestione dei riferimenti tramite radiocomando, alla comunicazione seriale con MATLAB, fino alla chiusura dell'anello di controllo in posizione. Per ciascuna fase sono state evidenziate le criticità emerse e le scelte adottate per superarle.

4.1. Accenni allo Sviluppo dello Yaw

Il controllo dell'angolo di imbardata (yaw) non rientrava tra gli obiettivi del progetto, ma ha costituito il punto di partenza del lavoro. Nonostante ciò, si è scelto di riportare i principali passaggi di sviluppo, poichè hanno messo in luce limiti meccanici della struttura di supporto che hanno condizionato le scelte successive.

Come per gli angoli di pitch e roll, il riferimento di yaw è stato inizialmente generato a partire dal segnale PWM del radiocomando e, solo in una fase successiva, trasmesso digitalmente da MATLAB tramite comunicazione seriale. La differenza sostanziale rispetto agli altri angoli risiede nel fatto che questi ultimi mantenevano il riferimento a un valore

fisso finché la levetta rimaneva spostata dalla posizione iniziale, mentre il riferimento di yaw è stato implementato in forma incrementale, secondo la legge

$$\psi_{ref}(k+1) = \psi_{ref}(k) + (\delta(k) - \delta_m) \cdot A \cdot v \cdot \Delta t \tag{4.1}$$

dove k indica un generico istante di campionamento, $\delta(k)$ è il duty cycle ricevuto dal radiocomando all'istante k, δ_m è il duty cycle corrispondente alla posizione centrale della levetta, A è l'ampiezza dell'incremento (ovvero la variazione del riferimento in corrispondenza della massima escursione della levetta), $\Delta t = 5\,\text{ms}$ è il passo di campionamento del ciclo di controllo e v è un fattore di velocità che regola la pendenza del riferimento. Dall'Equazione 4.1 si osserva che, spostando la levetta dalla posizione centrale, l'angolo di riferimento viene incrementato. Quando la levetta viene riportata in posizione centrale, l'incremento si arresta e il riferimento rimane sul valore raggiunto.

Sulla base di questo riferimento è stata condotta la sintesi e la taratura del regolatore di yaw, inizialmente implementato nella forma PID completa, comprendente le azioni proporzionale, integrale e derivativa. Le prove sperimentali hanno tuttavia evidenziato i limiti fisici del supporto lungo l'asse di imbardata. In particolare, l'attrito e i vincoli meccanici richiedevano coppie elevate, portando l'uscita del regolatore a lavorare frequentemente in saturazione.

Tale condizione era ulteriormente aggravata dal margine di sicurezza particolarmente restrittivo imposto all'escursione del duty cycle dei motori. In queste condizioni, l'azione derivativa, opponendosi alla velocità angolare del velivolo, riduceva ulteriormente l'autorità di controllo disponibile e la reattività del sistema. Si è pertanto scelto di adottare un regolatore PI, eliminando il contributo derivativo e consentendo al sistema di seguire con maggiore efficacia il riferimento di yaw, anche a velocità angolari più elevate. La legge di controllo adottata è quindi:

$$u_\psi(t) = K_p\, e_\psi(t) + K_i \int_0^t e_\psi(\tau)\, d\tau$$

Alla luce di queste limitazioni meccaniche, si è scelto di utilizzare esclusivamente gli angoli di pitch e roll per l'inseguimento della traiettoria, mantenendo lo yaw su riferimento costante.

4.2. Ricezione dei Dati da MATLAB

Sulla base di queste considerazioni, il primo intervento sul codice ha riguardato lo scambio di dati tra l'ambiente MATLAB e la scheda STM32. È stato sviluppato lo script `invia_dati`, con il compito di trasmettere, per un intervallo di tempo prefissato, il parametro `mode` e i tre riferimenti angolari di roll, pitch e yaw, codificati come stringhe ASCII formattate. Le stringhe ricevute dalla scheda tramite la porta seriale vengono quindi interpretate per estrarre i singoli valori, che vengono utilizzati al posto di quelli precedentemente forniti dal radiocomando. Questa impostazione consente di automatizzare l'avvio e il pilotaggio del drone direttamente dalla postazione di lavoro, riducendo la necessità di intervento manuale da parte dell'operatore.

Il radiocomando è stato quindi riconfigurato come dispositivo di sicurezza, in modo da consentire lo spegnimento di emergenza dei motori in qualsiasi momento. Se la levetta è in posizione bassa la scheda segue i comandi provenienti da MATLAB; con la levetta in posizione intermedia viene forzato l'armo dei motori; e infine, con la levetta in posizione alta viene imposto lo spegnimento dei motori, indipendentemente dalla modalità ricevuta da MATLAB. La logica di selezione della modalità è riportata in Figura 4.1, dove `mode` indica la modalità fornita da MATLAB e `radio_mode` quella fornita dal radiocomando. Con il trasmettitore spento, la scheda non riceve alcun comando valido dal radiocomando e può continuare a eseguire quelli di MATLAB. La perdita del segnale radio non è quindi gestita come condizione di emergenza, e il comportamento del ricevitore in failsafe non è stato verificato. La gestione di questa condizione è indicata tra gli sviluppi futuri.

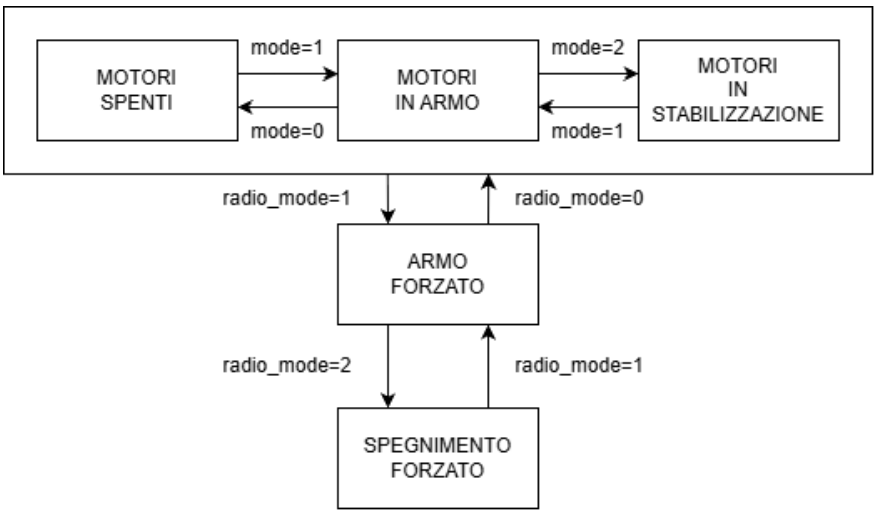


Figura 4.1. Logica di selezione della modalità operativa tra MATLAB e radiocomando.

4.3. Utilizzo della Traiettoria a Jerk Minimo

Una volta resa disponibile la comunicazione tra MATLAB e la scheda STM32, il passo successivo è stato lo sviluppo di un codice in grado di trasformare uno spostamento desiderato sugli assi x e y in una sequenza di riferimenti angolari di pitch e roll. Il modello adottato è la **traiettoria a jerk minimo**, introdotta nel paragrafo 2.4.2, che minimizza l'integrale del quadrato del jerk (derivata temporale dell'accelerazione) e garantisce profili di posizione, velocità e accelerazione continui, evitando sollecitazioni impulsive sulla struttura meccanica.

Nota 4.1 Per uno spostamento desiderato, il modello di traiettoria permette di determinare l'accelerazione di riferimento nel tempo. Nell'ipotesi di hovering e di piccoli angoli, l'accelerazione traslazionale può essere approssimata attraverso gli angoli di assetto secondo le relazioni

$$\ddot{x} \simeq g\,\theta, \qquad \ddot{y} \simeq -g\,\phi, \tag{4.2}$$

dove θ e ϕ rappresentano rispettivamente gli angoli di pitch e roll, mentre g è l'accelerazione di gravità. Da queste relazioni vengono quindi ricavati i riferimenti angolari necessari per ottenere lo spostamento desiderato.

Lo script MATLAB riceve in ingresso lo spostamento lineare richiesto, calcola la corrispondente sequenza di riferimenti di pitch e roll e la trasmette, campione per campione, alla scheda STM32. I riferimenti ricevuti vengono quindi utilizzati dai rispettivi regolatori PID di assetto per comandare il velivolo.

La configurazione ottenuta opera tuttavia in catena aperta rispetto alla posizione. Infatti, la scheda STM32 utilizza la retroazione dell'assetto misurato dall'IMU, ma non dispone di una misura diretta della posizione del velivolo. Di conseguenza, la risposta sul piano x-y è influenzata dalle incertezze del modello, dagli errori nella stima dell'assetto e dai disturbi esterni. Tali errori possono accumularsi nel tempo e causare derive nella posizione, per cui il moto ottenuto risulta qualitativamente coerente con il riferimento, ma non è ancora privo di errore a regime. Lo schema a blocchi della catena aperta ottenuta è riportato in Figura 4.2.

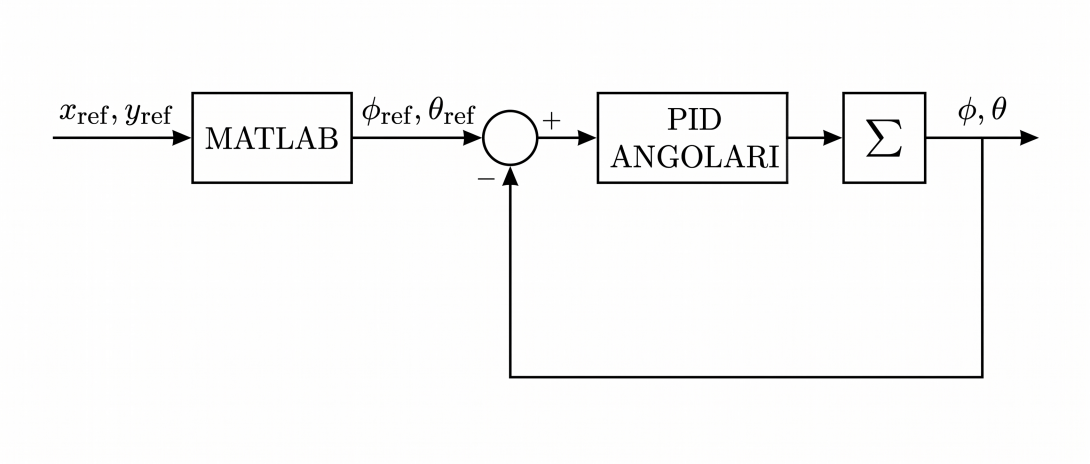


Figura 4.2. Schema di controllo in catena aperta rispetto alla posizione.

Per valutare a posteriori il moto compiuto è stato sviluppato lo script `ricostruzione_traiettoria`, che applica il modello inverso agli angoli misurati dall'IMU, ovvero, integra due volte le accelerazioni stimate tramite l'Equazione 4.2 per ottenere velocità e posizione.

Nota 4.2 La doppia integrazione rende la stima particolarmente sensibile al rumore e agli offset presenti nelle misure, che possono causare una progressiva deriva della posizione stimata. Per ridurre questo effetto, i dati angolari sono stati filtrati mediante un filtro a media mobile su N campioni,

$$\bar{\theta}(k) = \frac{1}{N} \sum_{i=0}^{N-1} \theta(k-i),$$

in modo da attenuare le componenti ad alta frequenza. La stima, pur presentando imprecisioni, è risultata sufficientemente affidabile per valutare il moto del velivolo.

4.4. Implementazione della Macchina a Stati Finiti

Validata la catena di comunicazione tra MATLAB e STM32, si è proceduto a predisporre la struttura necessaria al controllo in catena chiusa. In questa fase la retroazione sulla posizione non è ancora attiva, ma viene definita un'architettura software in grado di garantire un'esecuzione deterministica e sicura della manovra.

La prima modifica ha riguardato la modalità di trasmissione dei riferimenti. Nella configurazione precedente MATLAB inviava il riferimento istante per istante, esponendo il

sistema alle latenze e al jitter della comunicazione seriale. Nella nuova configurazione, invece, MATLAB trasmette l'intera traiettoria angolare prima dell'esecuzione. La scheda STM32 memorizza i riferimenti in appositi array e, una volta verificata la completezza del trasferimento, li applica ai PID di assetto un campione per ciclo di controllo. In questo modo la temporizzazione dei riferimenti non dipende più dalla comunicazione seriale, ma è determinata dal ciclo di esecuzione del microcontrollore.

Per gestire la sequenza operativa è stata adottata una macchina a stati finiti (FSM, *Finite State Machine*), che gestisce il caricamento della traiettoria, l'armo dei motori, la fase di decollo e stabilizzazione in hovering, l'inseguimento della traiettoria e la successiva fase di atterraggio. Le transizioni tra gli stati avvengono al verificarsi di condizioni definite, come il completamento del trasferimento dei dati o la scadenza di un intervallo temporale prefissato. In questo modo il comportamento del sistema risulta sequenziale e prevedibile. La logica di sicurezza del radiocomando, descritta in precedenza, rimane inoltre attiva durante tutte le fasi della procedura. Gli stati della macchina sono riportati in Tabella 4.1 e rappresentati in Figura 4.3.

Stato	**Denominazione**	**Funzione**
0	Attesa	Sistema inizializzato, in attesa dei dati da MATLAB
1	Caricamento	Ricezione e memorizzazione della traiettoria
2	Pronto	Trasferimento completato e verificato
3	Armo	Abilitazione dei motori
4	Hovering iniziale	Stabilizzazione su riferimento nullo per un tempo prefissato
5	Inseguimento	Applicazione della traiettoria caricata
6	Hovering finale	Stabilizzazione post-manovra per un tempo prefissato
7	Disarmo	Spegnimento dei motori in sicurezza

Tabella 4.1. Stati della macchina a stati finiti.

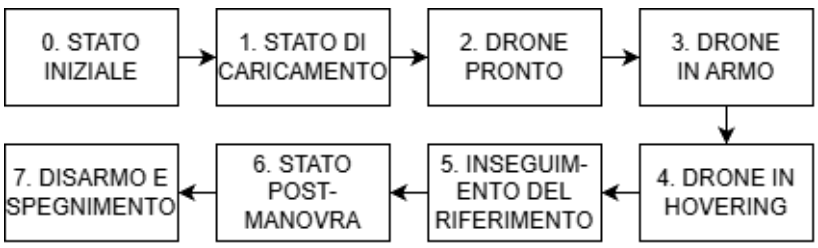


Figura 4.3. Diagramma della macchina a stati finiti.

Una volta che il codice MATLAB viene avviato, il sistema si trova nello stato 0. Durante la ricezione della traiettoria il sistema passa allo stato 1 e, a trasferimento completato, viene portato allo stato 2. In seguito si passa all'armo dei motori (stato 3) e ad una fase di hovering su riferimento nullo (stato 4), di durata prefissata, che consente al drone di stabilizzarsi prima della manovra e di partire da condizioni iniziali note. Terminato l'hovering, il drone insegue la traiettoria ricevuta (stato 5) e, a percorso concluso, effettua una seconda fase di hovering (stato 6), per assestare il sistema al termine della manovra. Infine i motori vengono disarmati e spenti (stato 7), concludendo così il volo. Questa struttura costituisce la base sulla quale integrare la stima di posizione e chiudere l'anello di controllo.

4.5. Controllo in Catena Chiusa

Poste le basi strutturali per il controllo in catena chiusa, l'ultima fase del lavoro ha riguardato la chiusura dell'anello di controllo in posizione. Come primo tentativo è stato impiegato un classico regolatore PID, che riceve in ingresso l'errore tra posizione di riferimento e posizione stimata e restituisce direttamente il riferimento angolare da inviare all'anello di assetto, secondo lo schema di Figura 4.4.

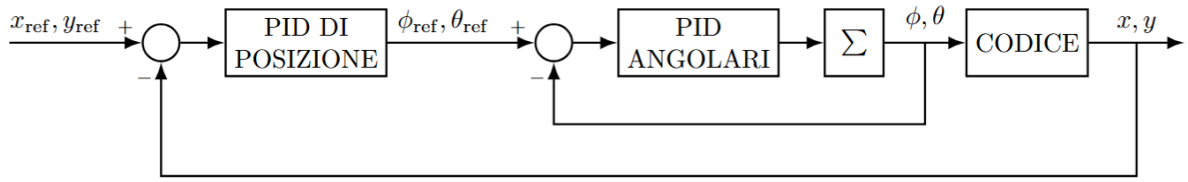


Figura 4.4. Schema di controllo in catena chiusa con regolatore PID in posizione.

Questa soluzione si è rivelata poco efficace, siccome la risposta in posizione mostrava ritardi non trascurabili, che persistevano a prescindere dalla taratura dei guadagni. Il motivo è dato dal fatto che, come indica l'Equazione 4.2, il legame tra angolo di assetto e posizione è quello di un doppio integratore, posto in cascata alla dinamica dell'anello di assetto. Il PID, agendo solo sull'errore, interviene soltanto dopo che questo si è manifestato e, mancando di un modello matematico alla base, non è in grado di tradurre in modo rigoroso lo scostamento di posizione negli angoli di pitch e roll necessari a compensarlo.

Si è quindi deciso di adottare un controllore del secondo ordine, basato sul modello matematico analizzato nella sezione 2.5. L'accelerazione desiderata viene calcolata come somma di un termine pari all'accelerazione della traiettoria di riferimento, e di una correzione proporzionale agli errori di posizione e di velocità:

$$\ddot{x}_{des} = \ddot{x}_{ref} + 2\zeta\omega_n \left(\dot{x}_{ref} - \dot{\hat{x}}\right) + \omega_n^2 \left(x_{ref} - \hat{x}\right), \tag{4.3}$$

dove $\hat{x}$ e $\dot{\hat{x}}$ sono la posizione e la velocità stimate. Un'espressione analoga vale per l'asse y. Le accelerazioni desiderate vengono poi convertite nei riferimenti di pitch e roll invertendo l'Equazione 4.2.

Nota 4.3 Supponendo che l'anello di assetto sia sufficientemente più rapido di quello di posizione, sostituendo l'Equazione 4.3 nel modello si ottiene per l'errore $e = x_{ref} - \hat{x}$ la dinamica

$$\ddot{e} + 2\zeta\omega_n \dot{e} + \omega_n^2 e = 0,$$

ovvero quella di un sistema del secondo ordine asintoticamente stabile per $\zeta > 0$ e $\omega_n > 0$.

In questo modo l'accelerazione di riferimento permette al drone di anticipare le accelerazioni richieste dalla traiettoria, eliminando il ritardo osservato con il PID, mentre i termini di retroazione compensano disturbi e incertezze di modello [15]. Grazie a questa formulazione, il sistema ha raggiunto la stabilità necessaria per inseguire la traiettoria con un errore significativamente ridotto, soddisfacendo l'obiettivo fissato per il progetto. Lo schema di controllo finale è riportato in Figura 4.5.

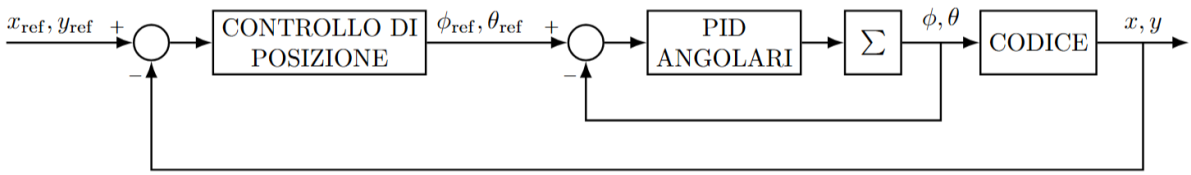


Figura 4.5. Schema di controllo in catena chiusa con controllore del secondo ordine.

Un ulteriore vantaggio di questo approccio è la semplicità di taratura, ridotta alla scelta dei soli parametri ζ e ω_n. Il coefficiente di smorzamento ζ regola la sovraelongazione della risposta, mentre la pulsazione naturale ω_n ne determina la prontezza, così che i due aspetti possano essere tarati in modo indipendente.

5

Sviluppo e Simulazioni Software

In questo capitolo verranno approfondite le operazioni che vengono effettuate dalle varie componenti software del sistema.

5.1. Comunicazione tra MATLAB e Microcontrollore

5.1.1. Collegamento Fisico e Parametri della Linea

Lo scambio di dati tra il calcolatore e la scheda avviene attraverso la periferica USART3 del microcontrollore STM32H745, che sulla scheda di sviluppo è instradata verso il Virtual COM Port del programmatore ST-LINK. Dal lato PC il collegamento è quindi visto come una comune porta seriale, gestita in MATLAB tramite la funzione `serialport`. La linea è configurata a 115200 baud, con 8 bit di dato, nessuna parità, un bit di stop e senza controllo di flusso. La banda utile risultante è di circa 11,5 kB/s per direzione.

5.1.2. Scelta del Protocollo Testuale

Per lo scambio dei dati si è adottato un protocollo testuale in codifica ASCII, senza impacchettamento binario, ovvero, ogni messaggio viene inviato come riga di testo leggibile, con i campi separati da spazi e terminata dal carattere di fine riga (`LF`). Dal lato MATLAB il

terminatore è impostato con `configureTerminator(s,"CR/LF","LF")`. Le righe ricevute dalla scheda terminano con `CR/LF`, quelle inviate con il solo `LF`.

Nota 5.1 Questa scelta richiede un numero di byte maggiore rispetto a una codifica binaria, ma offre diversi vantaggi: i messaggi possono essere letti e inviati manualmente tramite un qualsiasi terminale seriale; inoltre, non dipende da come i numeri in virgola mobile sono rappresentati in memoria nè dall'ordine dei byte. Infine, se una riga risulta corrotta, può essere scartata senza perdere la sincronizzazione della comunicazione, poiché il terminatore permette di individuare l'inizio del messaggio successivo.

5.1.3. Ricezione a Interrupt della Coda di Righe

La ricezione dei dati avviene tramite interrupt, tramite la funzione `HAL_UART_Receive_IT`. A ogni byte ricevuto viene invocata la callback `HAL_UART_RxCpltCallback`, che inserisce il carattere ricevuto nel buffer utilizzato per comporre la riga e, quando viene ricevuto il terminatore, completa la stringa e la copia in una coda di righe complete. Infine riarma la ricezione del byte successivo. Queste operazioni vengono eseguite in tempi trascurabili, così da ridurre al minimo l'interferenza degli interrupt sul resto del codice.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  if (huart->Instance == USART3) {
    if (mat_rxByte == '\n' || mat_rxByte == '\r') {
      if (mat_rxIdx > 0) {
        uint8_t next = (mat_qHead + 1) % MAT_QUEUE_N;
        if (next != mat_qTail) { /* coda non piena */
          mat_rxLine[mat_rxIdx] = '\0';
          memcpy(mat_queue[mat_qHead], mat_rxLine, mat_rxIdx + 1);
          mat_qHead = next;
        }
      }
      mat_rxIdx = 0;
    }
    else if (mat_rxIdx < MAT_RX_LEN - 1)
      mat_rxLine[mat_rxIdx++] = (char)mat_rxByte;
    else
      mat_rxIdx = 0; /* riga troppo lunga */
    HAL_UART_Receive_IT(&huart3, &mat_rxByte, 1);
  }
}
```

L'elaborazione effettiva delle righe avviene all'interno del ciclo principale, nella funzione `mat_processLine`, che svuota la coda a ogni iterazione. Questa separazione tra la callback, che scrive l'indice di testa, e il consumatore, che legge quello di coda, riduce al minimo la durata dell'interrupt e non ritarda gli altri interrupt del sistema, in particolare quelli dei timer che gestiscono il controllo e acquisiscono il radiocomando. La coda, di 16 righe da 64 caratteri, sostituisce una prima implementazione basata su un buffer singolo con flag di disponibilità, che perdeva le righe quando arrivavano più rapidamente di quanto il ciclo principale riuscisse a elaborarle. In caso di errore di ricezione (ad esempio un *overrun*) la callback `HAL_UART_ErrorCallback` serve ad azzerare il buffer e riarmare la ricezione, evitando che la periferica resti bloccata.

5.1.4. Elaborazione delle Stringhe

Il riconoscimento delle variabili numeriche è affidato alle funzioni della libreria standard `strtol` e `strtof`. Entrambe restituiscono, tramite il secondo argomento, un puntatore al primo carattere non convertito. Se tale puntatore coincide con quello di partenza, il campo non contiene un numero valido e la riga viene rifiutata. Una funzione ausiliaria viene utilizzata per saltare i separatori ammessi (spazio, virgola, punto e virgola, tabulazione).

```
long i = strtol(s, &e, 10); if (e == s) { printf("ERR␣FMT\r\n"); return; } s = e;
float x = strtof(s, &e); if (e == s) { printf("ERR␣FMT\r\n"); return; } s = e;
float y = strtof(s, &e); if (e == s) { printf("ERR␣FMT\r\n"); return; }
```

Rispetto a `sscanf`, questo approccio consente di verificare individualmente ciasun parametro e di assegnare le variabili di uscita solo dopo che l'intera riga è stata riconosciuta, così che un messaggio non valido non alteri i valori correnti. Tutte le variabili e le funzioni riguardanti la comunicazione con MATLAB sono state dichiarate con il prefisso `mat_`, per evitare conflitti con le macro già definite negli header del progetto.

5.1.5. Primo formato dei messaggi: invio in tempo reale

Nella prima versione del sistema MATLAB inviava un messaggio ogni $T_s = 20$ ms, contenente tutti i riferimenti da applicare in quell'istante:

```
<mode> <roll> <pitch> <yaw>
```

I quattro campi venivano memorizzati nelle variabili `mat_i1` (intero, modalità di volo: 0 arresto, 1 armamento, 2 stabilizzazione), `mat_roll` e `mat_pitch` (float, angoli di riferi-

mento in gradi, limitati a $\pm 10^{\circ}$) e `mat_f` (float, duty cycle del canale di imbardata, con la stessa codifica del radiocomando). In una delle versioni precedenti anche roll e pitch erano trasmessi come duty cycle interi e convertiti con la stessa logica utilizzata per il radiocomando. Il passaggio a valori reali in gradi è stato necessario per rappresentare i riferimenti continui prodotti dalle traiettorie descritte nella Sezione 5.2.

Il riferimento di yaw, essendo un comando di velocità, veniva integrato una volta per ogni riga ricevuta, con la stessa frequenza (50 Hz) con cui lo integrava la callback del radiocomando. La modalità di volo viene validata a partire dal segnale del radiocomando, che, svolgendo la funzione di arresto di emergenza, consente di eseguire la modalità richiesta da MATLAB se la levetta è in posizione bassa, mentre forza l'armo o l'arresto dei motori se la levetta si trova rispettivamente in posizione intermedia o alta.

Nota 5.2 Sul PC, l'invio dei dati avviene a intervalli regolari, introducendo una pausa che tiene conto del tempo già trascorso nel ciclo di esecuzione. All'istante k si attende

$$t_{\text{attesa}} = (k+1)\,T_s - t_{\text{trascorso}},$$

in modo che i ritardi dovuti all'esecuzione del ciclo non si accumulino nel tempo. Il trasferimento attraverso il sistema operativo e la USB introduce comunque una variabilità di alcuni millisecondi sugli istanti di arrivo, ma il microcontrollore la tollera mantenendo l'ultimo valore ricevuto fino all'arrivo del successivo.

5.2. Conversione della Traiettoria in Angoli di Riferimento

5.2.1. Profilo a Jerk Minimo

Le traiettorie nel piano orizzontale sono costruite con il profilo a jerk minimo, che tra due estremi di una traiettoria minimizza l'integrale del quadrato della derivata terza della posizione (equivalente alla bruschezza della traiettoria). Detto $\tau = t/T \in [0,1]$ il tempo normalizzato sulla durata T del tratto, la posizione lungo uno spostamento D è

$$x(t) = x_0 + D\,s(\tau), \qquad s(\tau) = 10\tau^3 - 15\tau^4 + 6\tau^5,$$

con velocità e accelerazione

$$\dot{x}(t) = \frac{D}{T}\,(30\tau^2 - 60\tau^3 + 30\tau^4), \qquad \ddot{x}(t) = \frac{D}{T^2}\,(60\tau - 180\tau^2 + 120\tau^3).$$

Velocità e accelerazione sono nulle agli estremi del tratto, condizione necessaria perché il drone possa partire e arrivare in quiete. L'accelerazione raggiunge il valore massimo in $\tau = (3 \pm \sqrt{3})/6$, pari a

$$|\ddot{x}|_{\max} = \frac{10}{\sqrt{3}} \frac{D}{T^2}.$$

Inizialmente, per i percorsi a punto di interesse (*waypoint*) lo stesso profilo veniva applicato a ciascun tratto, con gli spostamenti lungo x e y scalati dalla stessa funzione $s(\tau)$, così che ogni tratto risultasse rettilineo.

5.2.2. Passaggio alle Traiettorie Continue

Col passaggio alle traiettorie continue, si è presentato un ulteriore problema che ha trasformato l'impiego di τ come variabile temporale da una scelta facoltativa a una necessità. Questo perchè, per una generica traiettoria $\mathbf{p}(u) = [f_x(u), f_y(u)]^T$, scegliendo come percorso la circonferenza è possibile esprimere la traiettoria nel seguente modo:

$$\mathbf{p}(u) = \begin{bmatrix} R\cos(\omega u) \\ R\sin(\omega u) \end{bmatrix}$$

A differenza della precedente traiettoria a punti di interesse, per la quale velocità e accelerazione si annullavano agli istanti iniziale e finale, in questo caso tali condizioni non sono necessariamente soddisfatte. Per esempio, prendendo la velocità all'istante $u = 0$ si ottiene:

$$\mathbf{v}(0) = \left.\frac{d\mathbf{p}(u)}{du}\right|_{u=0} = \begin{bmatrix} -\omega R\sin(0) \\ \omega R\cos(0) \end{bmatrix} = \begin{bmatrix} 0 \\ \omega R \end{bmatrix} \neq \mathbf{0}$$

Il quadricottero non può partire a velocità non nulla, perchè all'istante iniziale servirebbero una spinta e un'accelerazione infinite. Per risolvere questo problema, si introduce una funzione scalare $u = \tau(t)$ di *time scaling* dove t rappresenta il tempo fisico reale e τ indica il tempo normalizzato su T.

La velocità di avanzamento della variabile temporale normalizzata è data da:

$$\dot{\tau}(t) = \frac{d\tau(t)}{dt}$$

Siccome il tempo τ avanza come una rampa, la sua derivata ha un andamento a gradino. Si assume quindi che, durante la fase di *hovering*, l'avanzamento della variabile sia unitario

($\dot{\tau}(t) = 1$), e all'inizio e al termine della traiettoria l'avanzamento della variabile sia nullo ($\dot{\tau}(t) = 0$). Esprimendo la traiettoria in funzione del tempo reale si ottiene:

$$\mathbf{p}(t) = \mathbf{f}(\tau(t))$$

Poichè $\mathbf{f}$ dipende da τ, che a sua volta dipende da t, per derivare occorre applicare la regola della composizione, in modo da ottenere

$$\mathbf{v}(t) = \frac{d\mathbf{p}(t)}{dt} = \frac{d\mathbf{f}(\tau)}{d\tau} \cdot \frac{d\tau}{dt}$$

Indicando con $\mathbf{f}'(\tau)$ la derivata prima di $\mathbf{f}(\tau)$ rispetto a τ, si ottiene che:

$$\mathbf{v}(t) = \mathbf{f}'(\tau) \cdot \dot{\tau}$$

con l'ipotesi di $\dot{\tau}(0) = 0$, si ottiene che all'istante iniziale:

$$\mathbf{v}(0) = \mathbf{f}'(\tau) \cdot 0 = \mathbf{0}$$

analogamente per l'istante finale. Il velivolo parte pertanto da fermo, e termina la traiettoria da fermo, indipendentemente dalla forma geometrica della funzione $\mathbf{f}$. Derivando nuovamente rispetto al tempo si ottiene l'accelerazione:

$$\mathbf{a}(t) = \frac{d\mathbf{v}(t)}{dt} = \frac{d}{dt}\left[\mathbf{f}'(\tau(t)) \cdot \dot{\tau}(t)\right]$$

Dalla derivata del prodotto si ottiene che:

$$\mathbf{a}(t) = \mathbf{f}''(t) \cdot \dot{\tau}^2 + \mathbf{f}'(\tau) \cdot \ddot{\tau}$$

il cui primo contributo rappresenta l'accelerazione centripeta, e il secondo indica l'accelerazione tangenziale. Quindi, con questa formulazione anche l'accelerazione si annulla agli istanti iniziale e finale della traiettoria.

Poichè il profilo di $\dot{\tau}$ è calcolato sul modello dello strappo minimo, le funzioni $\tau(t)$, $\dot{\tau}(t)$ e $\ddot{\tau}(t)$ risultano continue e differenziabili, garantendo riferimenti di posizione, velocità e accelerazione continui e facilmente inseguibili dai controllori interni. La geometria del percorso rimane quindi invariata rispetto al caso precedente, mentre cambia solo la legge temporale con cui viene percorso.

5.2.3. Utilizzo del Modello Inverso

Un velivolo multirotore può generare forza solo lungo il proprio asse verticale, mentre per un'accelerazione orizzontale deve inclinarsi, orientando la spinta lungo la direzione del vettore $\mathbf{a} = [a_x,\ a_y,\ g]^T$. Supponendo una variazione di yaw nulla e quota costante, gli angoli di riferimento risultano

$$\theta_{\text{ref}} = \text{atan2}(a_x,\ g), \qquad \varphi_{\text{ref}} = \arcsin\left(\frac{a_y}{\|\mathbf{a}\|}\right), \tag{5.1}$$

dove il pitch θ produce il moto lungo x e il roll φ quello lungo y. Per angoli piccoli entrambe le relazioni si riducono a $\theta \simeq a_x/g$ e $\varphi \simeq a_y/g$. Questa proprietà è valida per qualunque multirotore, siccome nella relazione non compaiono nè la massa nè la spinta del drone.

5.3. Trasmissione della Telemetria dalla Scheda

5.3.1. Utilizzo di `printf` sulla Seriale

I dati di telemetria vengono inviati dal microcontrollore a MATLAB come testo, sulla stessa USART3. Per farlo si usa la normale funzione `printf`. Nella libreria newlib, `printf` scrive un carattere alla volta chiamando la funzione `__io_putchar`. Nel progetto questa funzione è stata riscritta in modo che ogni carattere venga mandato sulla seriale:

```
int __io_putchar(int ch)
{
  HAL_UART_Transmit(&huart3, (uint8_t*)&ch, 1, 0xFFFF);
  return ch;
}
```

La trasmissione avviene in modalità bloccante (*polling*), ovvero il programma aspetta che ogni carattere sia stato trasmesso prima di andare avanti. La ricezione invece continua a funzionare tramite interrupt. Le due cose non si ostacolano, perché la libreria HAL tiene separati lo stato della trasmissione (`gState`) e quello della ricezione (`RxState`). Poichè la versione ridotta della libreria (newlib-nano), non stampa i numeri decimali di default, il collegamento del progetto richiede l'opzione `-u _printf_float`.

5.3.2. Frequenza e contenuto

La stampa parte solo quando il flag `flag_print` è attivo, e viene attivato dalla callback del timer TIM1 con cadenza nominale di 50 ms. La riga viene stampata dentro le funzioni `stopMotors`, `armingMotors` e `stabilize`, subito dopo la lettura dell'IMU. In questo modo tutti i valori della riga si riferiscono allo stesso istante. Il contenuto è riportato nella Tabella 5.1.

Campi	Grandezza	Unità
1–4	comandi ai quattro motori	% duty
5–6	rollio e beccheggio misurati	deg
7–10	ingressi virtuali (spinta, roll, pitch, yaw)	–
11–16	guadagni dei PID di rollio e beccheggio	–
17–18	riferimenti di rollio e beccheggio	deg
19–20	imbardata continua e suo riferimento	deg
21–22	posizione stimata $\hat{x}$, $\hat{y}$	m
23–24	posizione di riferimento x_{ref}, y_{ref}	m

Tabella 5.1. Campi della riga di telemetria.

Sulla stessa linea passano anche i messaggi di stato del protocollo (`READY`, `LOADED`, `START`, `RUN`, `ENDTRAJ`, `DONE`, `ERR`). Tutte le stampe partono dal ciclo principale e mai da un interrupt. Per questo le righe non possono mai interrompersi a vicenda.

5.3.3. Tempo di Trasmissione e Campionamento Effettivo

A 115200 baud, con 10 bit per carattere, una riga di circa 200 caratteri impiega circa 17 ms per essere trasmessa. La stampa è bloccante e avviene nel ciclo principale (la ricezione seriale è invece gestita in interrupt), che resta quindi fermo per questo periodo, bloccando il controllo d'assetto e allungando il periodo effettivo di telemetria. Tale periodo è stato ricavato sfruttando il fatto che la durata della manovra T_{traj} è nota. Se k_0 è il primo campione e k_1 l'ultimo in cui i riferimenti sono diversi da zero, allora:

$$T_c = \frac{T_{\text{traj}}}{k_1 - k_0}$$

Si ottiene $T_c \simeq 0{,}064$ s. Utilizzando il valore teorico di 50 ms, i dati non risultavano allineati con la traiettoria comandata. La macchina a stati e la stima di posizione usano

l'orologio di sistema `HAL_GetTick`, e non risentono di questa irregolarità. Per i PID dell'anello interno non è stata condotta una misura del periodo effettivo e del jitter, ma i guadagni sono stati tarati sperimentalmente nelle reali condizioni di esecuzione, per cui il comportamento riportato ne tiene implicitamente conto.

5.3.4. Lettura dei Dati in MATLAB

Lo script MATLAB legge la seriale in modo non bloccante. A ogni ciclo preleva i byte già arrivati (`NumBytesAvailable`), li aggiunge a un buffer e divide il buffer in righe usando il carattere di fine riga. L'ultimo pezzo, che può essere incompleto, viene tenuto da parte per il ciclo successivo. Le righe che corrispondono a messaggi di stato vengono gestite a parte. Le altre vengono lette con `sscanf` e salvate in un file CSV solo se contengono esattamente 24 valori, in modo che le righe tagliate o rovinate vengano scartate. Infine, la chiusura dei file e l'invio al drone del comando di arresto sono gestiti da una funzione `onCleanup`, che viene eseguita anche se lo script va in errore o viene interrotto manualmente.

5.4. Stima della Posizione del Drone

5.4.1. Modello

In assenza di un sensore di posizione, la posizione orizzontale viene stimata nel codice a partire dagli angoli misurati dall'IMU, invertendo l'Equazione 5.1:

$$\hat{a}_x = g \tan\theta, \qquad \hat{a}_y = \frac{g \tan\varphi}{\cos\theta},$$

e integrando due volte nel tempo le accelerazioni così ottenute. Il modello assume quota costante e yaw nullo.

5.4.2. Compensazione dell'Offset

Per ridurre gli errori di posizione causati da eventuali offset nella misura degli angoli, è stato introdotto un filtro per la stima dell'offset degli angoli misurati. Durante la fase di *hovering* prima della manovra, l'offset viene stimato a partire dalle misure acquisite e aggiornato progressivamente tramite una media mobile. All'inizio della manovra, il valore stimato viene utilizzato per correggere le misure successive, riducendo gli errori dovuti agli errori di stima e a eventuali inclinazioni residue.

5.4.3. Integrazione

L'integrazione usa lo schema di Eulero semi-implicito, con il passo Δt_k misurato dalla funzione di sistema (`HAL_GetTick`, con risoluzione pari a 1 ms) dal quale otteniamo:

$$\hat{v}_{k+1} = \hat{v}_k + \hat{a}_k \, \Delta t_k, \qquad \hat{p}_{k+1} = \hat{p}_k + \hat{v}_{k+1} \, \Delta t_k.$$

I passi con $\Delta t_k = 0$ (che corrispondono a due chiamate nello stesso millisecondo) vengono saltati e il tempo trascorso confluisce nel passo successivo. I passi anomali, con $\Delta t_k > 0{,}1$ s, vengono scartati. In questo modo la stima è indipendente dalla regolarità con cui viene eseguito il ciclo principale.

```
r -= est_offRoll; p -= est_offPitch; /* rimozione offset */
float ax = EST_G * tanf(p); /* pitch -> asse x */
float ay = EST_G * tanf(r) / cosf(p); /* roll -> asse y */
est_vx += ax * dt; est_vy += ay * dt; /* prima integrazione */
est_x += est_vx*dt; est_y += est_vy*dt; /* seconda */
```

All'inizio della manovra, l'integratore viene inizializzato a zero, in accordo con la traiettoria di riferimento, anch'essa definita in modo da partire dall'origine.

5.4.4. Ricostruzione della Traiettoria

La stima calcolata nel codice della scheda è stata confrontata con quella ricostruita a posteriori in MATLAB a partire dagli stessi angoli. Le due curve presentano la stessa forma e gli stessi punti di inversione. Lo scarto, che cresce nel tempo, è attribuibile alla diversa stima dell'offset e al diverso schema di integrazione.

Nota 5.3 Il metodo presenta un limite legato alla propagazione degli errori, poiché un errore angolare costante ε si traduce in un errore di accelerazione $e_a = g \tan \varepsilon$, che diventa

$$e_v(t) = e_a \, t, \qquad e_p(t) = \tfrac{1}{2} \, e_a \, t^2 \tag{5.2}$$

per velocità e posizione. La Tabella 5.2 riporta l'ordine di grandezza per $\varepsilon = 1^\circ$.

La stima ottenuta si basa quindi sul modello del moto (*dead reckoning*) e risulta affidabile su intervalli di tempo brevi, ma non può essere considerata equivalente a una misura diretta della posizione.

t [s]	e_v [m/s]	e_p [m]
5	0,86	2,1
10	1,71	8,6
20	3,42	34,2

Tabella 5.2. Errore di posizione dovuto a un errore angolare costante di 1° ($e_a \simeq 0{,}17$ m/s^2).

5.5. Controllo in Posizione a Ciclo Chiuso

5.5.1. Caricamento Anticipato della Traiettoria

Nella versione finale del sistema la traiettoria non viene più inviata in tempo reale, ma trasferita per intero al microcontrollore prima del volo. In questo modo i riferimenti seguono il clock della scheda e non risentono dei ritardi variabili dovuti al sistema operativo e alla comunicazione USB. Inoltre, una volta avviata la manovra, il volo non dipende più dal calcolatore. Alla scheda vengono inviate soltanto le posizioni di riferimento x e y, espresse in metri, mentre tutte le grandezze derivate vengono calcolate dalla scheda.

Il caricamento avviene attraverso la sequenza di messaggi indicata:

- `TRAJ <N> <dt_ms>`: dichiara il numero di punti e il passo temporale, e la scheda risponde con `READY`.
- `<i> <x> <y>`: un punto per riga. La scheda verifica che l'indice sia quello atteso, e in caso contrario risponde `ERR IDX <k>`, indicando il punto da cui riprendere.
- Ricevuto l'ultimo punto, la scheda calcola le derivate e risponde con `LOADED`.
- `GO` avvia la sequenza di volo; `STOP` la interrompe in qualsiasi momento.

La traiettoria viene salvata in vettori che contengono al massimo 2000 punti, il che, con un passo di 20 ms, corrisponde a una manovra di 40 s.

Velocità e accelerazione di riferimento vengono calcolate una sola volta, al termine del caricamento, tramite differenze centrate

$$v_k = \frac{x_{k+1} - x_{k-1}}{2\Delta}, \qquad a_k = \frac{x_{k+1} - 2x_k + x_{k-1}}{\Delta^2},$$

ponendo nulli i valori agli estremi, dove il drone è fermo. Come verifica intermedia, gli angoli ottenuti applicando il modello inverso dell'Equazione 5.1 a queste accelerazioni sono stati confrontati con quelli calcolati in precedenza in MATLAB, e le due serie di valori sono risultate identiche.

5.5.2. Macchina a Stati

La sequenza di volo è gestita tramite una macchina a stati, descritta nella Tabella 5.3. Il passaggio da uno stato all'altro e la durata delle diverse fasi sono determinati utilizzando il clock di sistema. In particolare, il tempo trascorso dall'ingresso nello stato corrente viene calcolato come differenza tra il tempo attuale e l'istante di ingresso. Durante la manovra, questo tempo viene utilizzato per individuare il punto della traiettoria da seguire. In questo modo, la durata delle fasi dipende dal tempo trascorso e non dal numero di iterazioni del ciclo di controllo, che può variare leggermente nel tempo.

Stato	Durata	Modalità	Azioni
`IDLE`, `LOAD`, `RDY`	–	arresto	attesa, caricamento, traiettoria pronta
`ARM`	2 s	armamento	azzeramento della stima
`HOV`	30 s	stabilizzazione	riferimenti nulli, stima dell'offset
`RUN`	T_{traj}	stabilizzazione	controllo di posizione
`AFT`	3 s	stabilizzazione	riferimenti nulli
`LND`	1 s	armamento	ritorno a `RDY`

Tabella 5.3. Stati della sequenza di volo.

Come accennato in precedenza, il radiocomando rimane attivo come dispositivo di sicurezza. Quando la levetta di arresto è azionata la macchina a stati viene riportata in `RDY` e i motori vengono fermati, indipendentemente dallo stato corrente. A ogni iterazione del ciclo principale prima viene effettuato il controllo di assetto, poi l'aggiornamento della stima, poi il controllo di posizione (solo in `RUN`), in seguito l'avanzamento della macchina a stati ed infine l'elaborazione dei messaggi ricevuti.

5.5.3. Legge di Controllo di Posizione

Supponendo che il controllo di assetto sia sufficientemente rapido da far seguire al drone l'angolo desiderato, ciascun asse orizzontale può essere approssimato come un sistema a

doppia integrazione, descritto da $\ddot{x} = a$. Si definisce quindi l'accelerazione desiderata

$$a_{\text{des}} = a_{\text{ref}} + K_d\,(v_{\text{ref}} - \hat{v}) + K_p\,(x_{\text{ref}} - \hat{x}), \tag{5.3}$$

dove x_{ref}, v_{ref} e a_{ref} sono posizione, velocità e accelerazione della traiettoria e $\hat{x}$, $\hat{v}$ le stime descritte nella Sezione 5.4. Definito l'errore $e = x_{\text{ref}} - x$ e sostituendo l'Equazione 5.3 nel modello, si ottiene

$$\ddot{e} + K_d\,\dot{e} + K_p\,e = 0,$$

cioè l'errore evolve come un sistema del secondo ordine la cui dinamica non dipende dalla traiettoria. Imponendo pulsazione naturale ω_n e coefficiente di smorzamento ζ:

$$K_p = \omega_n^2, \qquad K_d = 2\zeta\omega_n.$$

I guadagni del controllore sono definiti quindi in base alla dinamica desiderata per l'errore. Il termine a_{ref} corrisponde all'inversione del modello lungo la traiettoria nominale. Ponendo $\omega_n = 0$ la legge si riduce a $a_{\text{des}} = a_{\text{ref}}$, analogo al controllo in catena aperta, condizione utilizzata per verificare il corretto funzionamento dell'implementazione prima di chiudere l'anello di controllo.

L'accelerazione desiderata viene convertita negli angoli di riferimento per i controllori di assetto:

$$\theta_{\text{ref}} = \arctan\frac{a_{x,\text{des}}}{g}, \qquad \varphi_{\text{ref}} = \arctan\frac{a_{y,\text{des}}}{g}.$$

```
pos_errX = mat_refX - est_x;
float ax_des = mat_refAX + KD_X*(mat_refVX - est_vx) + KP_X*pos_errX;
if (ax_des > POS_ACC_MAX) ax_des = POS_ACC_MAX;
if (ax_des < -POS_ACC_MAX) ax_des = -POS_ACC_MAX;
pitchRef = MAT_SGN_PITCH * atanf(ax_des / MAT_G) * 57.29578f;
```

La legge di controllo non possiede uno stato interno (siccome mancano termini integrali e derivativi), per cui non dipende dal passo di esecuzione.

5.5.4. Scelta dei Parametri

La struttura è organizzata in cascata, con l'anello di posizione che fornisce i riferimenti all'anello interno. Questa struttura si basa sull'ipotesi che l'anello interno sia molto più rapido di quello esterno, ovvero, che la pulsazione dell'anello esterno sia almeno tre volte più bassa della banda dell'anello interno. Quest'ultima è stata stimata dal ritardo di inseguimento dell'assetto, misurata come $\omega_{\text{in}} \simeq 1/\tau$ (Tabella 5.4).

Asse	τ [s]	ω_{in} [rad/s]	ω_n [rad/s]	$\omega_{\text{in}}/\omega_n$
y (rollio)	0,7	1,4	0,32	4,5
x (beccheggio)	1,3	0,77	0,32	2,4

Tabella 5.4. Banda stimata dell'anello interno e rapporto con la pulsazione dell'anello esterno, per i parametri usati nelle prove.

Il ritardo dell'anello interno riduce lo smorzamento effettivo rispetto a quello nominale, tanto più quanto il rapporto tra le bande è ridotto. Questo spiega le sovraelongazioni osservate ai vertici del percorso, che risultano infatti più marcate lungo l'asse x.

5.6. Sintesi del Controllore PID

Nel Capitolo 2 è stato trattato il modello matematico ideato per controllare il volo del drone. In questa sezione verrà trattata la realizzazione di un regolatore PID, ovvero il controllore che è stato scelto per regolare il volo del drone dal punto di vista angolare.

5.6.1. Funzionamento del PID

Il controllore PID (proporzionale-integrale-derivativo) è un particolare regolatore industriale che presenta il vantaggio di limitare la sintesi di controllo alla scelta di alcuni parametri, che andranno poi a modificare l'azione totale del controllore [17]. Questi controllori vengono posti in contro-reazione nel sistema a ciclo chiuso, in modo che essi possano agire direttamente sull'errore tra l'uscita del sistema, indicata con $y(t)$, e il riferimento, che rappresenta il valore ideale dell'uscita, e viene indicato invece con $r(t)$.

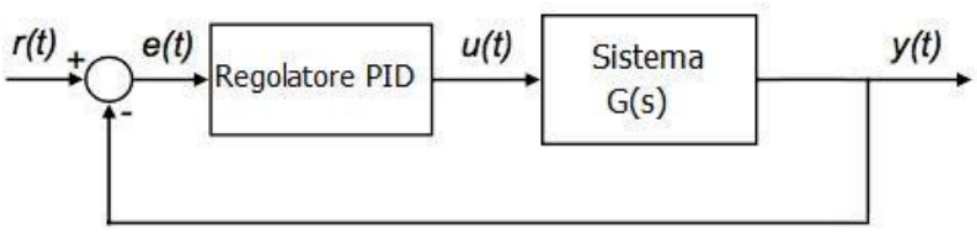


Figura 5.1. Schema a blocchi del PID

Nel sistema di controllo l'errore viene definito come $e(t) = r(t) - y(t)$. Nel caso in questione il riferimento è dato dall'angolo ideale che il drone deve raggiungere, mentre l'uscita è data dall'angolo registrato dal sensore IMU.

Il nome del regolatore PID deriva dal fatto che l'azione di controllo è data dalla somma di tre contributi, uno proporzionale all'errore, uno proporzionale all'integrale dell'errore, e uno proporzionale alla derivata dell'errore. È possibile descrivere l'azione di controllo tramite la seguente relazione:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt}$$

dove K_p, K_i e K_d rappresentano rispettivamente i guadagni dell'azione proporzionale, integrale e derivativa. La scelta di tali parametri influenza direttamente le prestazioni del sistema in termini di rapidità di risposta, stabilità e precisione nel raggiungimento del riferimento.

5.6.2. Tuning del Controllore

Il processo di scelta dei parametri del PID prende il nome di *tuning* (o taratura), ed è fondamentale per ottenere una buona risposta da parte del controllore, che si ottiene con un'adeguata combinazione delle tre risposte singole.

Solitamente l'azione derivativa viene utilizzata in combinazione con un altro metodo (P o PI), poichè agisce per contrastare la velocità di variazione della variabile di processo. Questa caratteristica rende l'azione derivativa ideale per alcuni utilizzi, ma la rende anche del tutto impraticabile per molti altri e inutilizzabile singolarmente, ed è per questo che si usa quasi sempre nelle applicazioni PID.

L'uso dell'azione derivativa può risultare molto vantaggioso, ma deve essere utilizzato nelle situazioni appropriate. Il presupposto iniziale dovrebbe essere quello di verificare se il processo funziona in modo soddisfacente anche solo con il controllo PI, poichè un'attenta regolazione dei fattori P e I può ottenere gran parte degli stessi benefici. Perciò l'aggiunta del termine derivativo deve essere effettuata con cautela e con un filtraggio appropriato. La risposta al variare della configurazione è riportata in Figura 5.2.

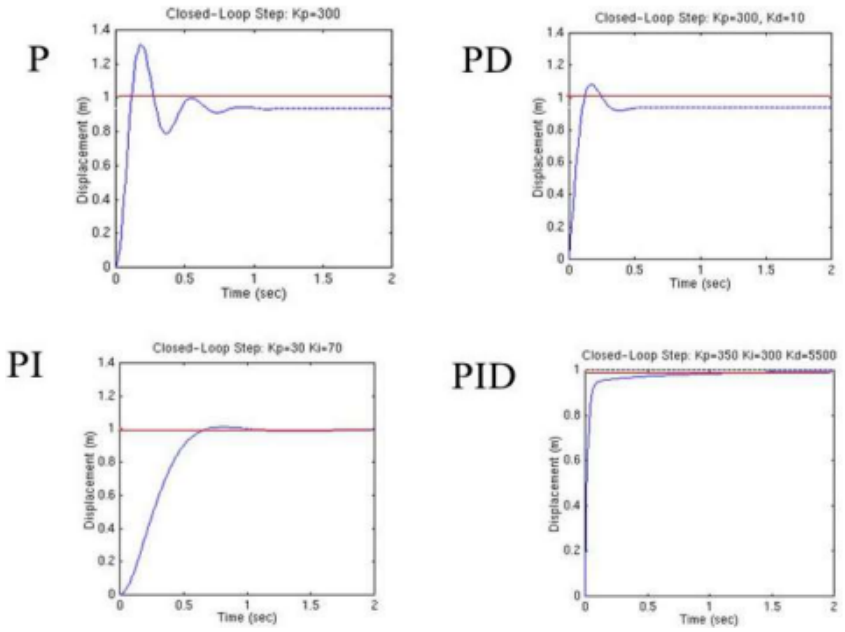


Figura 5.2. Combinazioni P, PD, PI e PID

La scelta del regolatore dipende dalle caratteristiche del processo e dal comportamento desiderato del sistema. Quando non è richiesto un controllo troppo specifico si può utilizzare un semplice controllore P; il controllore PI può essere utilizzato quando è necessario accelerare il tempo di assestamento e serve ridurre l'offset; il controllore PD quando serve accelerare il periodo transitorio e ci sono problemi di oscillazione e il PID per quando serve un controllo più completo che richiede l'azione di tutti e tre i controllori. Per quanto riguarda il controllo angolare, è stato utilizzato il controllo PID per intero, e il metodo di taratura utilizzato è il metodo di *trial and error*, che prevede una ricerca dei parametri per tentativi ed errori, sulla base di numerosi test e analisi delle risposte precedenti.

5.6.3. Discretizzazione del PID

Finora, è stato considerato il PID solamente nella sua formulazione a tempo continuo, mentre per implementarlo come algoritmo di controllo in un microcontrollore, è necessario definirlo come sistema a tempo discreto. Pertanto, occorre ricavare una formulazione discreta del PID a partire dalla sua rappresentazione continua [6].

La discretizzazione di un segnale è il processo con cui si passa da un segnale continuo a una rappresentazione discreta, definita solo in un insieme discreto di istanti di tempo. La discretizzazione di un segnale continuo consiste nel determinarne il valore a intervalli

di tempo regolari. L'intervallo tra due campionamenti consecutivi è detto **periodo di campionamento** ed è indicato con T. Quindi, per trovare l'andamento dell'errore $e(t)$ in un intervallo di tempo discreto, occorre trovare i valori di $e[k] = e(k \cdot T)$, dove k è un valore intero positivo.

Si nota che il termine proporzionale ha una discretizzazione immediata, siccome dato dal prodotto tra un guadagno e l'errore a tempo continuo:

$$u(t) = K_p \cdot e(t) \quad \Rightarrow \quad u[k] = K_p \cdot e[k]$$

Il termine integrale può essere discretizzato in diversi modi, tra questi nel progetto è stato selezionato il metodo di Eulero in avanti, ovvero un metodo per approssimare l'integrale dell'errore (corrispondente all'area sotto la curva dell'errore) ad una somma di rettangoli di base T e altezza pari al valore dell'errore all'estremità sinistra di ciascun intervallo.

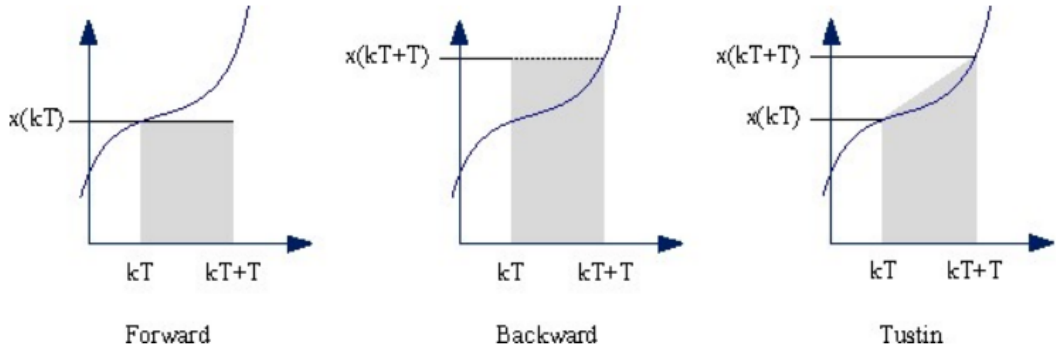


Figura 5.3. Metodi di approssimazione del termine integrale

Quindi in questo caso il valore dell'errore in ciascun intervallo di campionamento viene approssimato come una costante pari all'estremità sinistra.

$$u[k] = K_i \cdot \sum_{j=1}^{k} \{(kT + T) - kT\} \cdot e[j-1] = K_i \cdot \sum_{j=1}^{k} T \cdot e[j-1]$$

possiamo riscrivere la sommatoria (fino a k) come la sommatoria fino al termine $k-1$ sommata all'ultimo termine, cioè:

$$u[k] = K_i \cdot \sum_{j=1}^{k} T \cdot e[j-1] = \left[K_i \cdot \sum_{j=1}^{k-1} T \cdot e[j-1] \right] + K_i \cdot T \cdot e[k-1] = u[k-1] + K_i \cdot T \cdot e[k-1]$$

in questo modo diventa possibile scrivere l'output di controllo integrale in maniera ricorsiva senza calcolare la sommatoria fino al termine k per ogni istante di tempo.

Infine, per quanto riguarda il termine derivativo, si utilizza un'approssimazione basata sul rapporto incrementale, che teoricamente si potrebbe applicare sia in avanti che all'indietro, tuttavia, l'unica differenziazione possibile nella pratica risulta quella all'indietro, in quanto quella in avanti richiederebbe di conoscere un valore futuro del segnale.

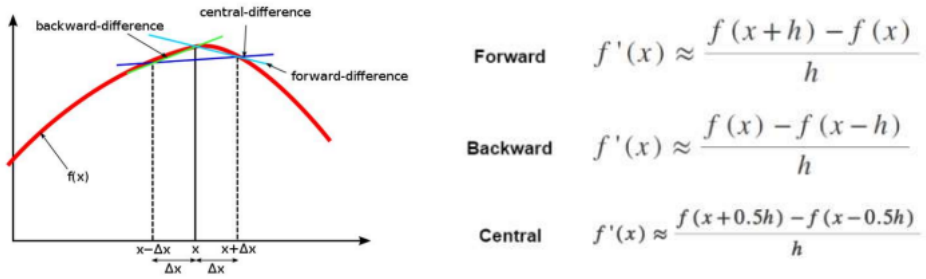


Figura 5.4. Metodi di approssimazione del termine derivativo

Pertanto, la derivata dell'errore viene approssimata in ogni intervallo di campionamento come una costante pari alla pendenza della retta secante tra l'errore corrente e l'errore precedente.

$$u[k] = K_d \cdot \frac{e[k] - e[k-1]}{T}$$

È stata quindi ricavata con successo un'implementazione digitale del controllore PID, in quanto basta sommare questi tre termini per ottenere l'azione di controllo totale.

5.6.4. Modifiche Attuate

Prima di descrivere l'implementazione tramite codice del controllore, è necessario illustrare alcune modifiche apportate alla struttura del PID, che si sono rivelate necessarie in seguito ad alcuni problemi riscontrati durante i test sperimentali del drone.

Problema del Wind-Up Integrale

Il problema del *wind-up* (letteralmente "caricamento eccessivo") è uno degli effetti collaterali che possono presentarsi nei controllori con azione integrale, e si verifica quando lo sforzo di controllo supera i limiti fisici del sistema.

Quando si verifica questa condizione, il sistema si trova in condizioni di saturazione. In una situazione del genere, l'azione di controllo effettivamente applicata al sistema non può aumentare ulteriormente, nonostante l'eventuale presenza di un errore. Se il termine

integrale continua ad essere aggiornato, esso accumula progressivamente l'errore, dando origine al fenomeno di *integral windup*. Per questo motivo, quando si è in condizioni di saturazione, il codice impedisce l'aggiornamento del termine integrale, siccome quest'ultimo contribuirebbe ad aumentare ulteriormente la saturazione. Questa soluzione prende il nome di *anti wind-up*.

Inseguimento dell'Uscita

Un altro cambiamento che è stato applicato al PID consiste nel fatto che il termine derivativo insegue solamente l'uscita e non l'errore tra riferimento ed uscita. Questo perchè, con lo spostamento del riferimento tramite radiocomando (problema che è stato trasmesso anche con il percorso digitale da MATLAB), l'errore presentava una variazione a gradino a causa del riferimento variabile. Siccome un ingresso a gradino presenta un contenuto in frequenza elevato e il termine derivativo tende ad amplificare tutto ciò che riceve in ingresso, questo comportava evidenti problemi nella risposta del controllore.

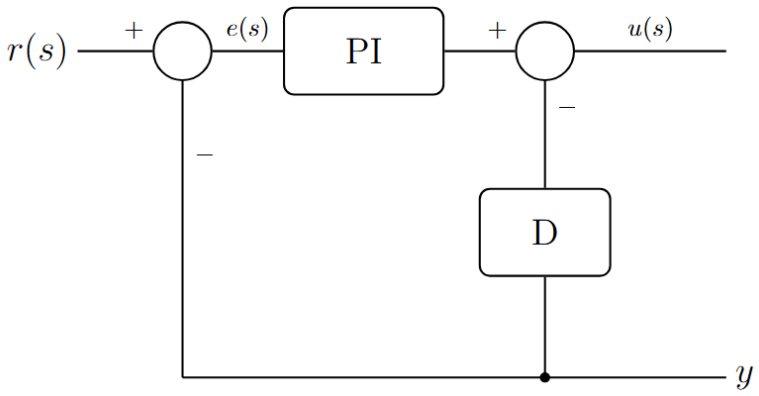


Figura 5.5. PID con derivatore in contro-reazione

Per questo motivo si è deciso di calcolare il termine derivativo sulla base dell'uscita prodotta dal sensore IMU e non più sull'errore col riferimento. Siccome l'output del derivatore è definito come

$$u[k] = \frac{K_d}{T} \cdot \Delta e[k]$$

e la differenza dell'errore è definita come

$$\Delta e[k] = r - y - r + y' = -y + y' = -\Delta y[k]$$

allora per modificare l'uscita del termine derivativo in modo che insegua l'uscita e non l'errore è necessario un cambiamento di segno:

$$u[k] = -\frac{K_d}{T} \cdot \Delta y[k]$$

Filtraggio con Media Mobile

Per limitare gli effetti dell'amplificazione del rumore ad alta frequenza introdotta dall'azione derivativa, gli ingressi utilizzati per il calcolo del termine derivativo sono stati filtrati mediante un filtro passa-basso. Questa scelta è stata applicata a causa dell'elevata sensibilità del drone alle componenti ad alta frequenza del segnale.

Inizialmente è stato applicato un filtro di tipo esponenziale, ovvero un filtro che dà peso a tutti i campioni passati con peso che decresce in maniera esponenziale, ma si è rivelato di scarsa efficacia. Pertanto, la scelta finale è ricaduta su un filtro a media mobile, che considera un numero finito di campioni.

Nota 5.4 L'uscita all'istante k, che rappresentiamo con $y[k]$, sarà data dalla media aritmetica degli ultimi N campioni dell'ingresso registrato $x[k]$:

$$y[k] = \frac{1}{N} \sum_{i=0}^{N-1} x[k-i]$$

I valori vengono memorizzati in una finestra (*array*) di dimensione fissa, e ogni nuovo campione inserito comporta l'eliminazione di quello più vecchio.

6

Test e risultati

Il presente capitolo è dedicato all'analisi dei risultati sperimentali ottenuti nel corso del progetto. Durante lo sviluppo sono state eseguite numerose prove, e, coerentemente con lo sviluppo descritto nel capitolo 4, si è deciso di riportare solamente quelle ritenute più significative, ovvero quelle che documentano il passaggio da una fase di sviluppo alla successiva e che consentono di confrontare i modelli teorici adottati con il comportamento effettivo del drone.

6.1. Condizioni di Partenza

Il lavoro è partito dalla taratura dei regolatori PID di assetto ereditata dal progetto precedente, dedicato alla rotazione attorno all'asse di yaw. I tre PID angolari di roll, pitch e yaw risultavano già calibrati e hanno costituito la base per implementare la gestione remota dei riferimenti.

I risultati di ciascuna prova sono rappresentati tramite figure MATLAB composte da quattro grafici, la cui struttura è rimasta invariata fino all'introduzione dei test in catena chiusa [11, 14]:

1. il primo grafico riporta i valori di duty cycle forniti ai motori, e permette di verificare l'eventuale raggiungimento dei limiti di saturazione imposti;

2. il secondo grafico riporta l'inseguimento dei riferimenti di pitch e roll da parte degli angoli misurati dall'IMU;

3. il terzo grafico riporta gli output dei tre PID di roll, pitch e yaw;

4. il quarto grafico riporta l'inseguimento dell'angolo di yaw rispetto al proprio riferimento.

Un esempio di questa rappresentazione, relativo alle condizioni di partenza, è riportato in Figura 6.1.

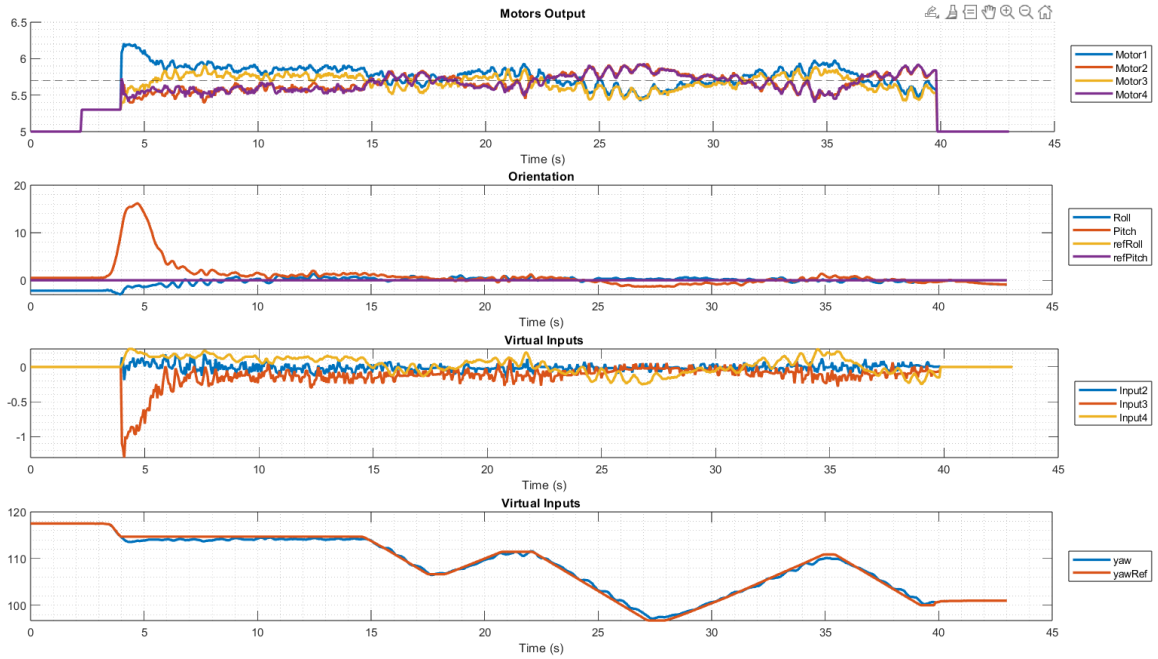


Figura 6.1. Risposta del sistema nelle condizioni di partenza, con i regolatori di assetto ottenuti dal progetto dello yaw.

Nel corso del progetto la calibrazione dei regolatori PID ha seguito un andamento sostanzialmente lineare, ad eccezione di alcune anomalie specifiche, ritenute trascurabili in quanto relative al movimento sull'angolo di yaw.

6.2. Primo Tentativo Invio Riferimenti da MATLAB

I primi test successivi all'implementazione della comunicazione seriale tra MATLAB e STM32 hanno evidenziato una risposta anomala e particolarmente brusca del quadricottero. La causa è stata individuata in un errore di temporizzazione, siccome l'aggiornamento dei riferimenti forniti ai PID avveniva all'interno di una routine di codice non appropriata, eseguita a una frequenza diversa da quella di campionamento dell'anello di controllo. Il riferimento veniva quindi modificato in modo asincrono rispetto all'azione di controllo, producendo variazioni a gradino tra un campione e il successivo. Tali discontinuità, amplificate in particolare dall'azione derivativa dei regolatori, determinavano una reazione fortemente irregolare del controllore, compromettendo la risposta complessiva prima che l'anello di retroazione potesse compensarla.

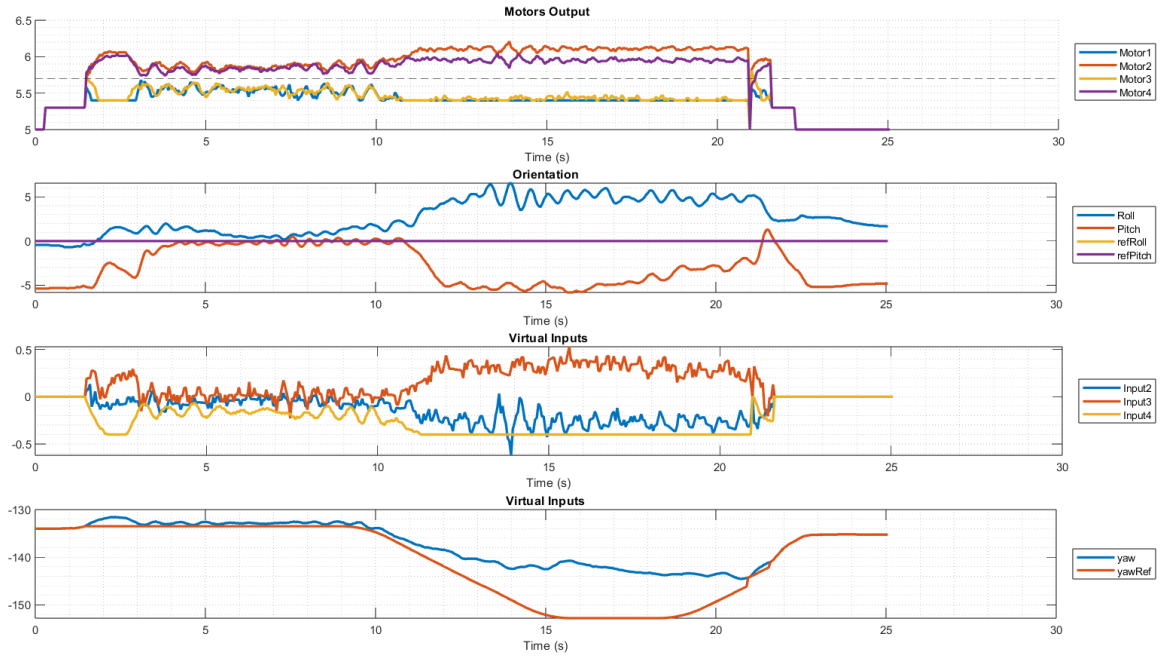


Figura 6.2. Test n°4: risposta del sistema in presenza dell'errore di temporizzazione nell'aggiornamento dei riferimenti.

Dalla Figura 6.2 si può notare il peggioramento della stabilità. Il primo grafico mostra che i duty cycle dei motori sono stati portati ripetutamente al limite di sicurezza, inducendo come conseguenza anche la saturazione del PID di yaw, come visibile nel terzo e quarto grafico. L'effetto della saturazione si è manifestato anche durante le prove sperimentali, con movimenti bruschi e oscillazioni marcate della struttura.

6.3. Correzione Invio Riferimenti da MATLAB

Le prove successive hanno permesso di risalire all'origine del problema. La correzione è consistita nello spostare l'aggiornamento dei riferimenti all'interno della stessa routine di codice temporizzata che esegue il calcolo dei regolatori, in modo da sincronizzare la frequenza di aggiornamento del riferimento con quella dell'anello di controllo e ridurre i fenomeni di instabilità.

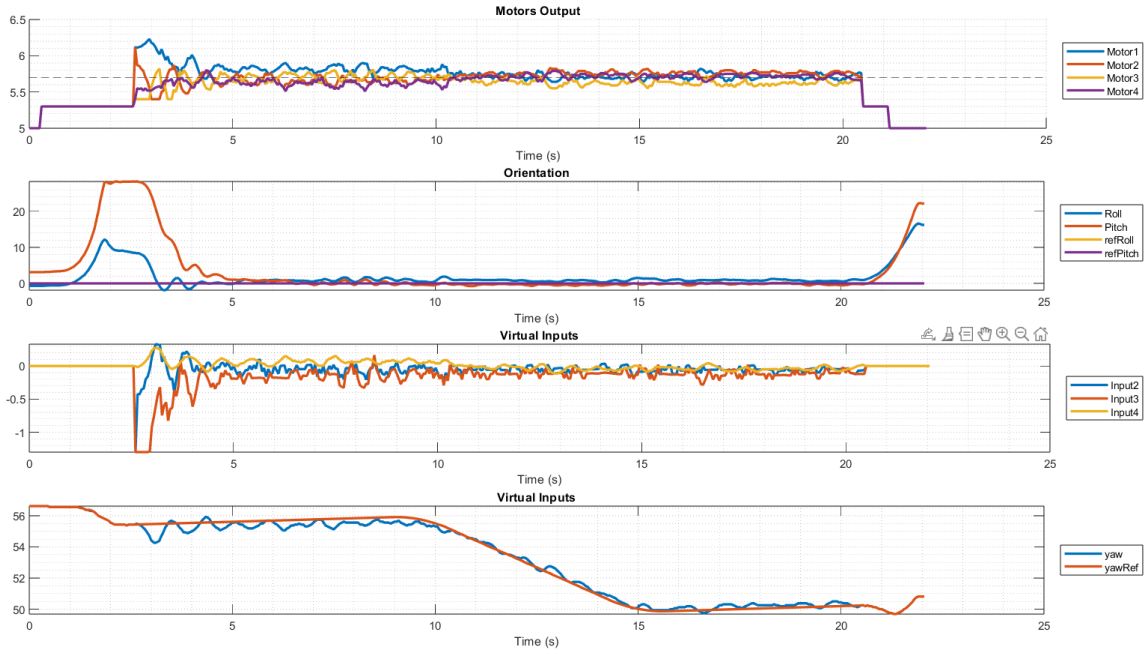


Figura 6.3. Test n°11: risposta del sistema dopo la correzione della temporizzazione.

Come mostrato in Figura 6.3, l'inseguimento dei riferimenti angolari è tornato a essere paragonabile a quello delle condizioni di partenza. L'errore di inseguimento si mantiene entro 1° per gli angoli di pitch e roll, mentre sull'angolo di yaw si osserva un errore iniziale di circa 2°, che a regime si riduce a circa 0.5°.

Per completezza, in questa fase è stato trasmesso da MATLAB anche il riferimento di yaw, allo scopo di verificare che il sistema fosse in grado di gestire la risposta del drone sull'asse più impegnativo in termini di sforzo di controllo anche tramite comunicazione remota. Per l'angolo di yaw è stata replicata la logica del radiocomando, inviando un valore simulato di duty cycle anzichè il valore angolare. Un duty superiore al valore medio δ_m produce quindi un incremento progressivo del riferimento, mentre il ripristino

al valore medio annulla l'incremento e mantiene il riferimento sull'ultimo valore raggiunto, senza riportarlo al valore iniziale.

6.4. Assetto da MATLAB in Catena Aperta

Raggiunto un buon livello di stabilità con i riferimenti inviati da remoto, il passo successivo è stato l'introduzione della traiettoria a jerk minimo per verificare l'inseguimento in catena aperta. In questa fase la conversione tra spostamento lineare e assetto avveniva interamente in MATLAB, mentre la scheda STM32 si limitava a ricevere i riferimenti di roll, pitch e yaw e ad applicarli ai regolatori di assetto.

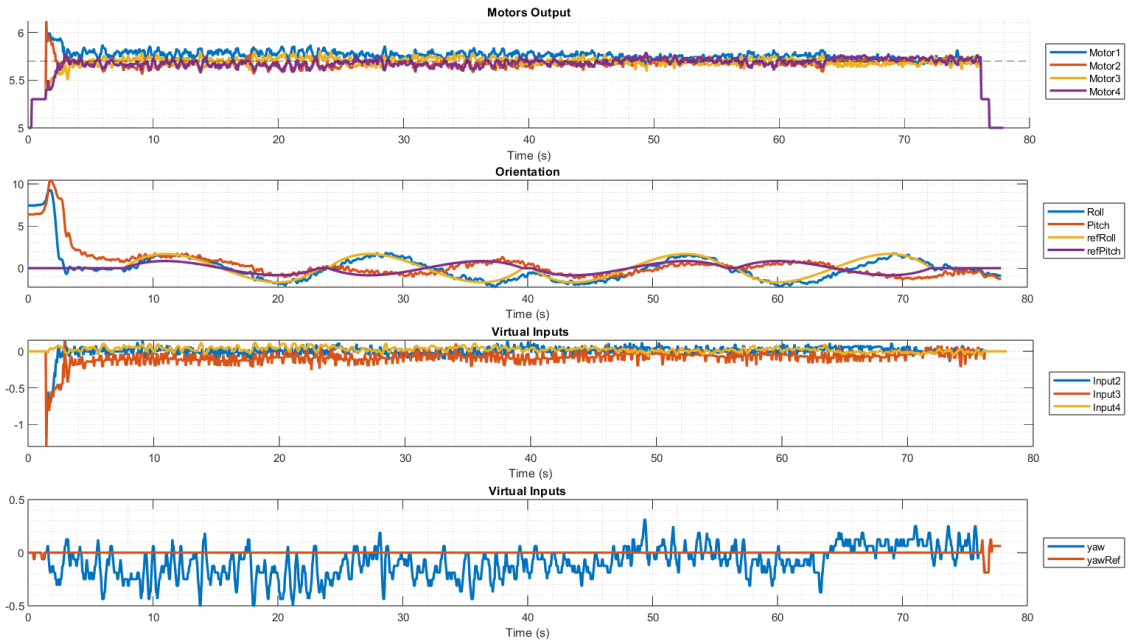


Figura 6.4. Test n°28: inseguimento in catena aperta dei riferimenti angolari ricavati da una traiettoria a jerk minimo.

Trattandosi di un controllo in catena aperta rispetto alla posizione, inizialmente l'unico indicatore disponibile per valutare l'esito della manovra era la fedeltà con cui gli angoli misurati dall'IMU inseguivano i riferimenti (secondo grafico della Figura 6.4). Successivamente, lo script `ricostruzione_traiettoria` ha permesso di visualizzare il percorso stimato del drone sovrapposto a quello ideale, applicando agli angoli misurati il modello inverso (4.2). Lo stesso algoritmo è stato in seguito integrato nel codice del microcontrollore.

È stato inoltre sviluppato lo script `calcolo_traiettoria`, che calcola in anticipo l'assetto di riferimento a partire dallo spostamento virtuale desiderato. Inizialmente lo script gestiva esclusivamente traiettorie punto-punto, e in seguito è stato esteso a traiettorie continue. La combinazione di questo script con quello dedicato alla ricostruzione della traiettoria consente di completare il processo di verifica, siccome `calcolo_traiettoria` genera i riferimenti angolari da fornire al drone, mentre `ricostruzione_traiettoria` utilizza gli spostamenti angolari misurati per stimare il percorso simulato seguito dal velivolo e confrontarlo con quello ideale. Il test n°32 illustra questo procedimento. In Figura 6.5 sono riportati gli angoli calcolati sulla base del percorso da seguire nel piano, visibile nel grafico in alto a destra.

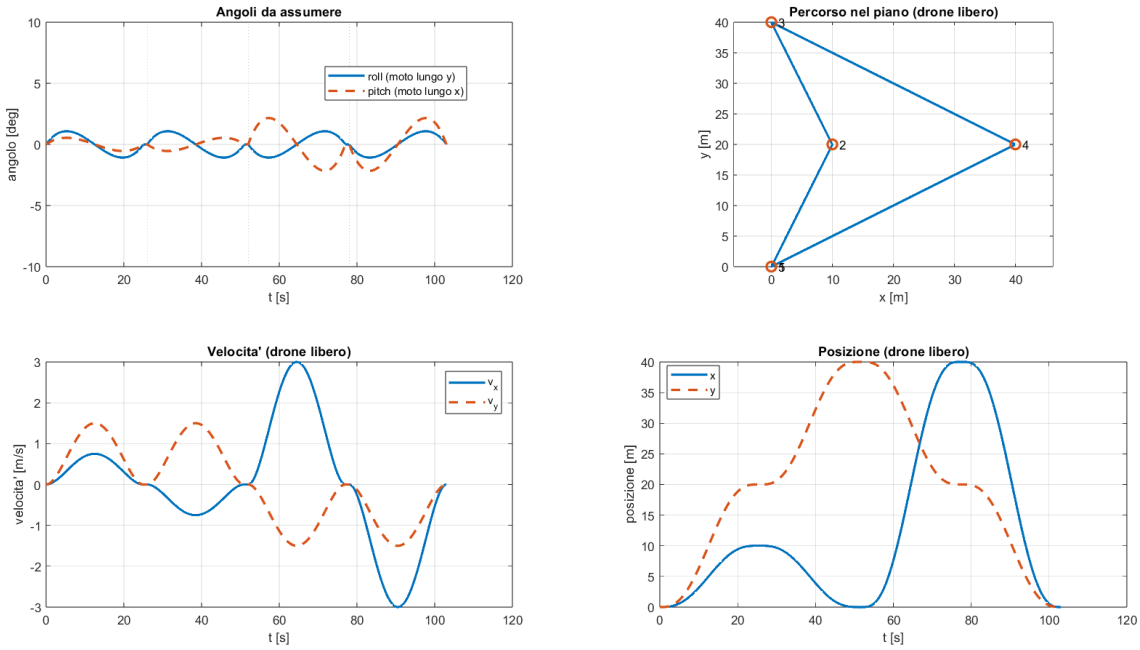


Figura 6.5. Test n°32: percorso nel piano e riferimenti angolari calcolati da `calcolo_traiettoria`.

La risposta del drone è riportata in Figura 6.6. I riferimenti di pitch e roll applicati coincidono con quelli calcolati in anticipo, a conferma del corretto invio della traiettoria, e risultano inseguiti fedelmente dagli angoli effettivi.

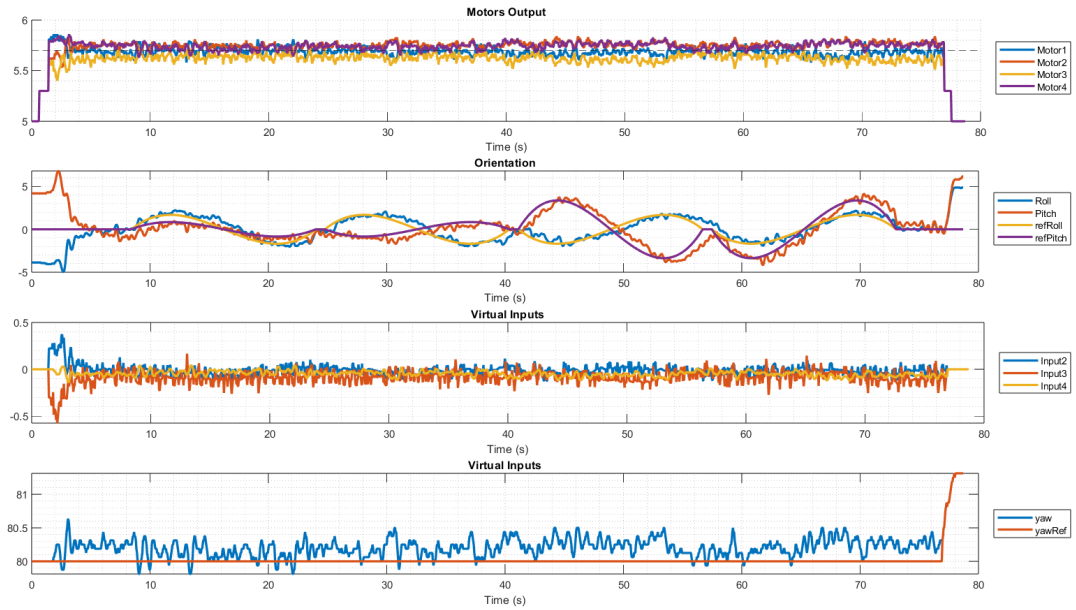


Figura 6.6. Test n°32: inseguimento dei riferimenti angolari.

A partire dagli angoli misurati, `ricostruzione_traiettoria` restituisce il percorso stimato, riportato in Figura 6.7. Lo scostamento rispetto al percorso ideale è dovuto al controllo in catena aperta, che, in assenza di correzioni, porta i piccoli errori di assetto ad accumularsi nel tempo, causando una deriva di posizione.

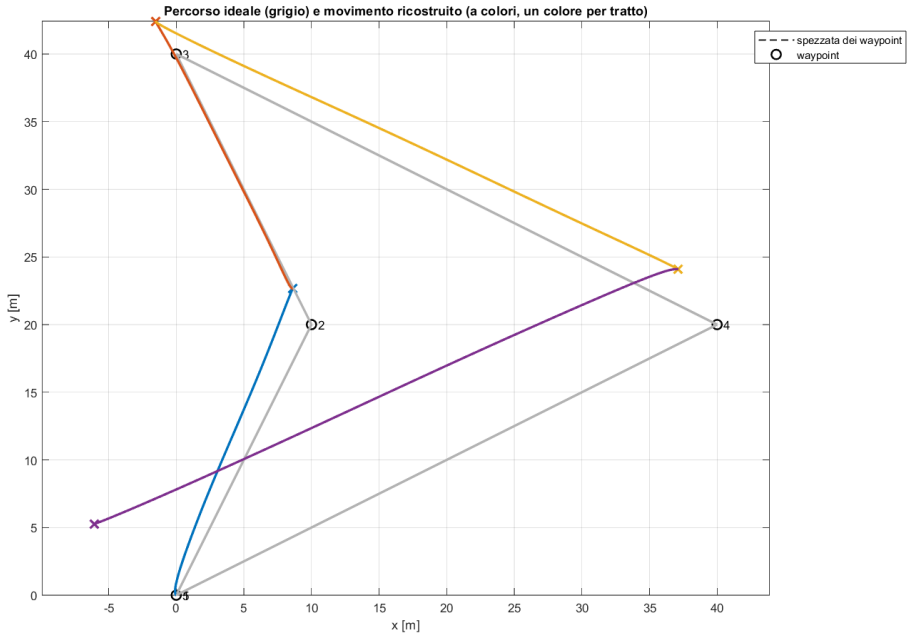


Figura 6.7. Test n°32: confronto tra percorso ideale e percorso ricostruito tramite `ricostruzione_traiettoria`.

6.5. Traiettoria da MATLAB in Catena Aperta

Consolidata la catena aperta, l'intera logica di conversione è stata trasferita nel codice della scheda STM32. La mappatura tra dinamica traslazionale e riferimenti di assetto viene ora eseguita nella scheda, mentre MATLAB è utilizzato unicamente per fornire il riferimento di posizione nel piano. Il microcontrollore calcola autonomamente l'assetto richiesto, pilota i motori e stima la posizione del velivolo, fornendo così i dati necessari per il controllo in catena chiusa.

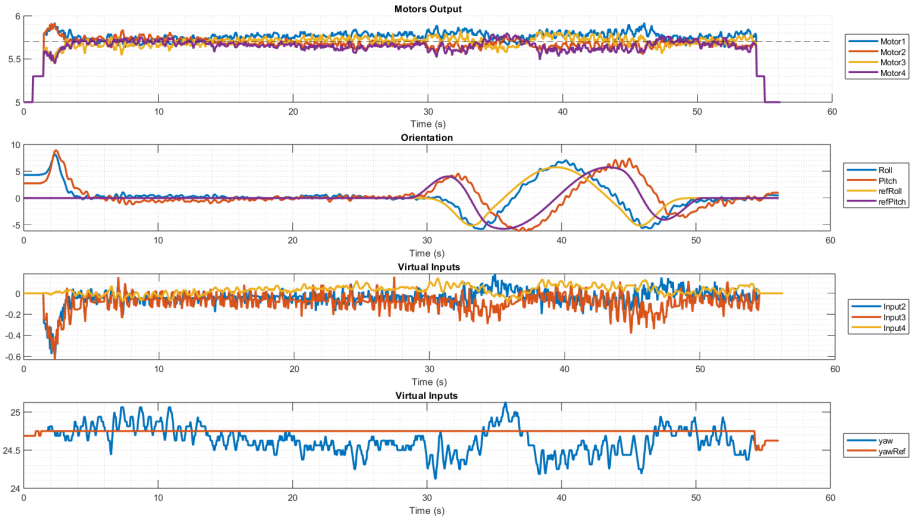


Figura 6.8. Test n°79: inseguimento in catena aperta con conversione posizione-assetto.

La Figura 6.9 riporta il percorso ottenuto da `ricostruzione_traiettoria` a partire dalla stima di posizione calcolata dalla scheda STM32. La correttezza della stima è fondamentale per il controllo in catena chiusa, che la utilizza come retroazione.

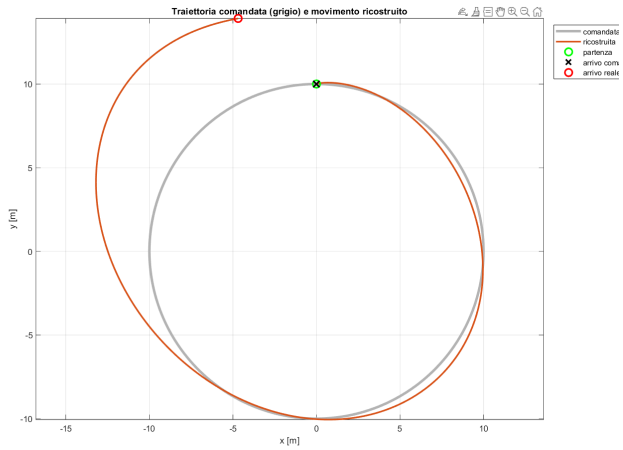


Figura 6.9. Test n°79: percorso ricostruito a partire dalla stima di posizione.

6.6. Circonferenza da MATLAB in Catena Chiusa

Con la stima di posizione calcolata dalla scheda e il controllore del secondo ordine dell'Equazione 4.3 implementato, sono iniziati i test di taratura dei parametri ζ e ω_n dei controllori relativi agli assi x e y. Nonostante le limitazioni fisiche degli attuatori, nelle prove condotte lo scostamento tra posizione virtuale stimata e traiettoria di riferimento è rimasto contenuto, come riportato nelle sezioni seguenti.

Il primo test di verifica dell'anello chiuso ha riguardato l'inseguimento di una traiettoria circolare. A partire da questo test, lo script `disegna_dati` sostituisce il quarto grafico, prima dedicato all'angolo di yaw e ormai non più significativo, con l'inseguimento in catena chiusa della posizione di riferimento da parte della posizione simulata.

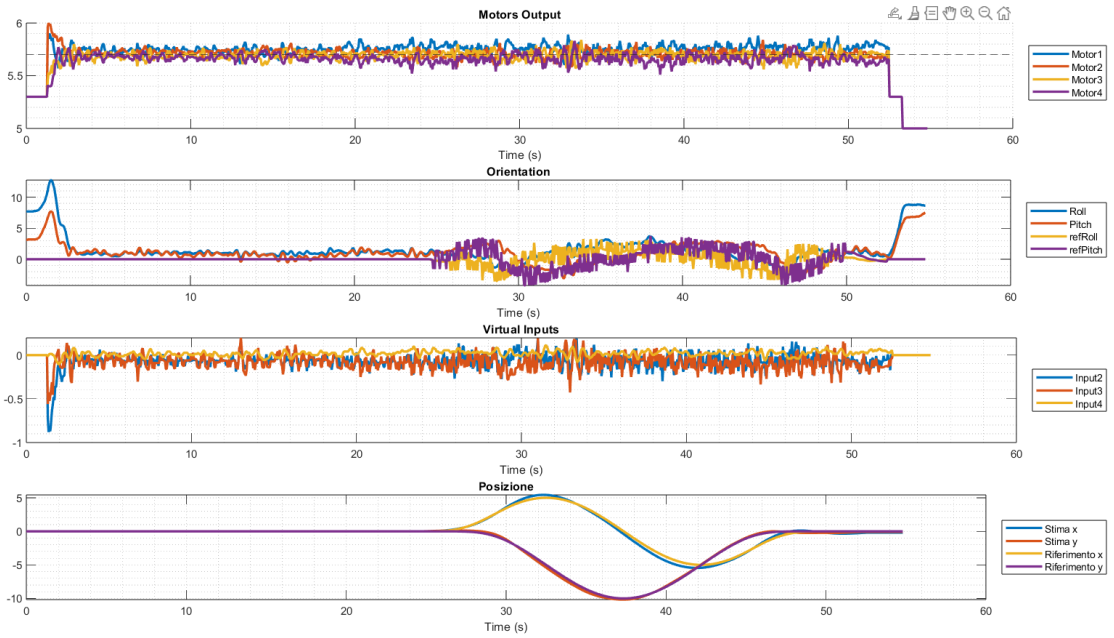


Figura 6.10. Test n°131: inseguimento in catena chiusa di una traiettoria circolare.

Nel quarto grafico della Figura 6.10 la stima di posizione sull'asse x (in blu) segue il riferimento (in giallo) con un errore massimo di 20–30 cm, mentre la stima sull'asse y (in rosso) segue il riferimento (in viola) con un errore massimo di circa 15 cm. Ne risulta un errore massimo complessivo di circa 30 cm sulla circonferenza, come visibile in Figura 6.11. La posizione simulata ha quindi completato il percorso circolare con uno scostamento contenuto rispetto al riferimento, verificando il funzionamento dell'anello chiuso.

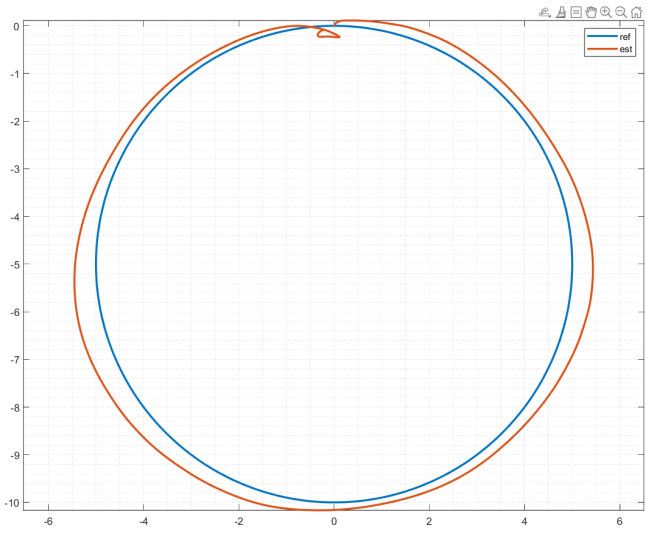


Figura 6.11. Test n°131: confronto nel piano tra circonferenza di riferimento e percorso stimato.

6.7. Rettilineo da MATLAB in Catena Chiusa

Un secondo test in catena chiusa ha previsto come riferimento un segmento rettilineo da percorrere in andata e ritorno. Questa traiettoria permette di valutare il comportamento del controllore in corrispondenza dell'inversione del moto, nella quale il drone deve arrestarsi e invertire la direzione di avanzamento.

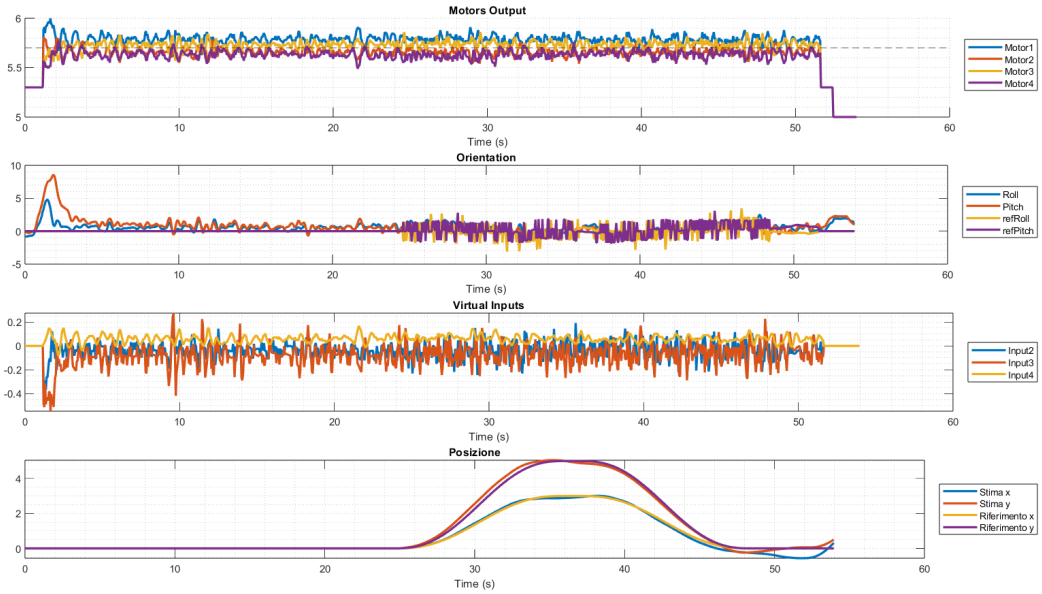


Figura 6.12. Test n°156: inseguimento in catena chiusa di una traiettoria rettilinea di andata e ritorno.

Come mostrato in Figura 6.12 e Figura 6.13, il riferimento non viene inseguito in modo perfetto, ma la posizione virtuale ricostruita si mantiene a una distanza dell'ordine dei 15 cm dal percorso di riferimento lungo l'intera manovra.

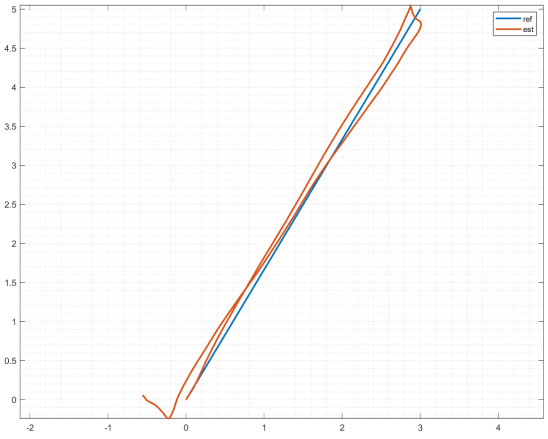


Figura 6.13. Test n°156: confronto nel piano tra traiettoria rettilinea di riferimento e percorso stimato.

6.8. Triangolare da MATLAB in Catena Chiusa

L'ultimo test in catena chiusa riportato ha riguardato un percorso triangolare, il più impegnativo tra quelli analizzati per la presenza di brusche variazioni di direzione in corrispondenza dei vertici.

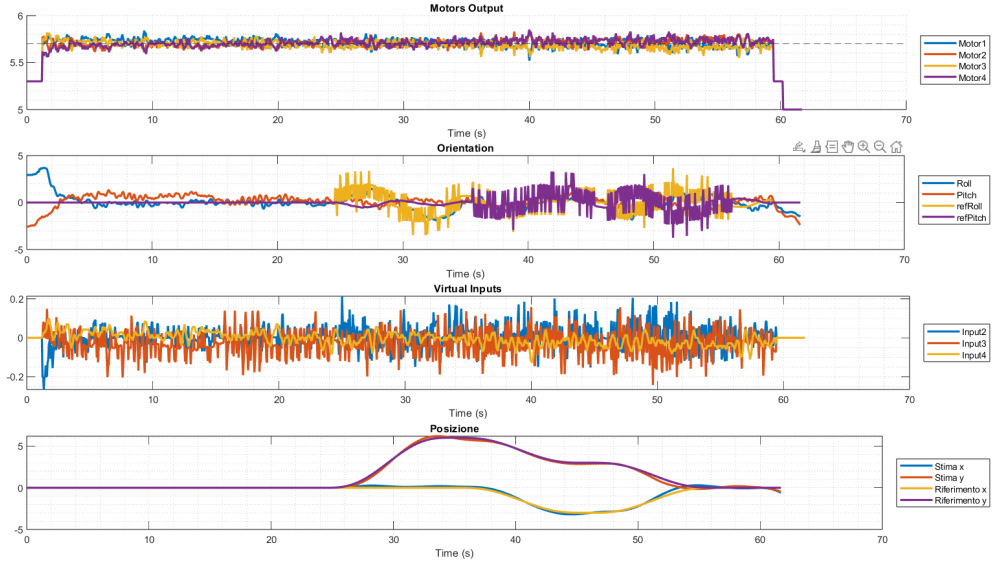


Figura 6.14. Test n°181: inseguimento in catena chiusa di una traiettoria triangolare.

Come si osserva in Figura 6.15, la posizione ricostruita perde precisione in corrispondenza dei vertici, dove la dinamica elevata richiesta dal cambio di direzione porta l'anello di controllo a smussare gli angoli del percorso. Lungo i lati, invece, lo scarto rispetto al riferimento è compreso tra 15 e 20 cm. Il tempo concesso al velivolo per completare la manovra è stato ridotto così da valutare la risposta del controllore in condizioni più impegnative. Nonostante questo, l'errore tra la traiettoria virtuale e quella nominale può ritenersi pienamente accettabile.

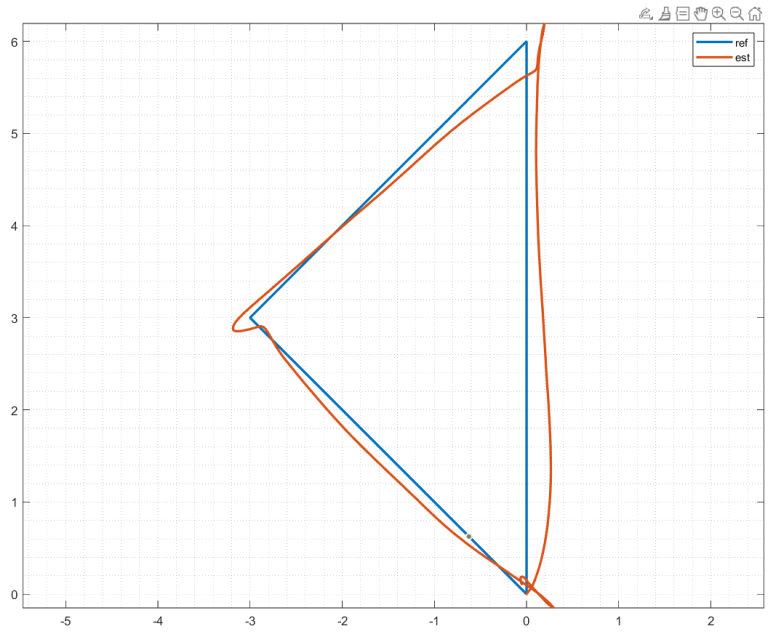


Figura 6.15. Test n°181: confronto nel piano tra traiettoria triangolare di riferimento e percorso stimato.

Nelle prove condotte, sono stati ottenuti i risultati migliori in corrispondenza di ω_n compreso tra 0.3 e 0.4 rad/s per entrambi gli assi di traslazione e ζ compreso tra 0.9 e 1.1, a seconda della specifica traiettoria da inseguire. Valori di ζ prossimi a 1 corrispondono a una risposta vicina allo smorzamento critico, priva di sovraelongazioni significative. I valori contenuti di ω_n derivano dalla necessità di limitare la banda dell'anello di posizione, così da tenerla separata da quella dell'anello di assetto e compatibile con i limiti degli attuatori. Questa scelta spiega anche lo smussamento osservato ai vertici del triangolo, siccome con una banda ridotta, l'anello di posizione non riesce a seguire i cambi di direzione troppo rapidi, che finiscono quindi per essere filtrati. Tali valori derivano da una taratura per tentativi e non da un'ottimizzazione sistematica.

Conclusioni e Sviluppi Futuri

L'attività di tirocinio presentata in questa relazione ha coperto l'intero ciclo di sviluppo di un sistema di controllo in retroazione per l'inseguimento di traiettorie nel piano. Rispetto al progetto di partenza, in cui i regolatori di assetto erano comandati esclusivamente da radiocomando, il lavoro ha introdotto la comunicazione seriale con MATLAB e la gestione remota dei riferimenti, compreso quello di yaw, l'esecuzione automatica dell'intera sequenza di manovra, la stima di posizione e l'anello di controllo dentro il codice della scheda, insieme agli strumenti MATLAB per la generazione delle traiettorie e per l'acquisizione e l'analisi dei dati sperimentali.

Le prove sono state condotte con il quadricottero vincolato a un supporto, che altera la dinamica rispetto al volo libero, e in assenza di un sensore di posizione. Pertanto, a differenza degli angoli di assetto che sono stati misurati dal sensore IMU, la posizione virtuale del drone è stata stimata dal codice per mezzo di una doppia integrazione. In queste condizioni l'anello chiuso ha mantenuto lo scarto tra traiettoria nominale e posizione ricostruita entro 15–30 cm nelle prove riportate. Tali risultati costituiscono una verifica funzionale sperimentale di una catena di elaborazione e controllo che comprende componenti fisici reali. Non esprimono però l'errore di posizionamento in volo libero, nè consentono di valutarne la ripetibilità, poichè per ciascuno scenario è riportata una singola prova.

Alla luce di questi risultati, si propongono alcuni sviluppi. Il primo è l'introduzione di una misura indipendente della posizione, a bordo o tramite un sistema esterno, che permetterebbe di validare la ricostruzione, sostituirla nella retroazione e passare a prove in volo libero, condotte in più ripetizioni. Il secondo consiste nel considerare lo spegnimento del radiocomando come una condizione di emergenza, configurando il failsafe del ricevitore e verificando nella codice la presenza del segnale radio. Il terzo riguarda la misurazione il periodo effettivo e il jitter dell'anello interno, eseguendo poi il controllo in un interrupt a periodo fisso e trasmettendo la telemetria in modo non bloccante. Infine, si potrebbe verificare la possibilità di ampliare l'intervallo di duty cycle dei motori, che si è rivelato limitante per la risposta dinamica del velivolo.

Bibliografia

[1] Karl Johan Åström and Tore Hägglund. *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America (ISA), Research Triangle Park, NC, 2nd edition, 1995.

[2] Karl Johan Åström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, 2008.

[3] Stuart Bennett. *A History of Control Engineering 1800–1930*. Peter Peregrinus, London, 1979.

[4] Hendrik W. Bode. *Network Analysis and Feedback Amplifier Design*. Van Nostrand, New York, 1945.

[5] Prof. Andrea Bonci. Slides del corso. Corso di Dynamics and Control of Intelligent Robots and Vehicles, 2025-2026.

[6] Prof. Andrea Bonci. Slides del corso. Corso di Laboratorio di Automazione, 2025-2026.

[7] Samir Bouabdallah. *Design and Control of Quadrotors with Application to Autonomous Flying*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, 2007.

[8] Eduardo F. Camacho and Carlos Bordons. *Model Predictive Control*. Springer, London, 2nd edition, 2007.

[9] Gianmarco Di Renzo. Studio e sviluppo di sistemi di controllo per diversi gradi di libertà di un drone, 2021-2022.

[10] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, April 2018.

[11] G. Masi, P. Orsini, D. Zeka. Stabilità del quadricottero, 2024-2025. Università Politecnica delle Marche, tech. rep., Corso di Laboratorio di Automazione.

[12] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[13] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2002.

[14] L. Di Stefano, M. Cretella, G. Mascioli et Al. Stabilità del quadricottero, 2025-2026. Università Politecnica delle Marche, tech. rep., Corso di Laboratorio di Automazione.

[15] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32, 2012.

[16] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525, Shanghai, China, 2011. IEEE.

[17] Davide Meneghel. Regolatori pid: tecniche di sintesi e problematiche implementative, 2010.

[18] Harry Nyquist. Regeneration theory. *Bell System Technical Journal*, 11(1):126–147, 1932.

[19] Abdel-Nasser Sharkawy. Minimum jerk trajectory generation for straight and curved movements: Mathematical analysis. In Sergey Y. Yurish, editor, *Advances in Robotics and Automatic Control: Reviews*, volume 2 of *Book Series*, chapter 5, pages 187–201. IFSA Publishing, Barcelona, Spain, 2021.

[20] Kimon P. Valavanis and George J. Vachtsevanos, editors. *Handbook of Unmanned Aerial Vehicles*. Springer, Dordrecht, 2015.

[21] George Zames. Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *IEEE Transactions on Automatic Control*, 26(2):301–320, 1981.