UNIVERSITÀ
POLITECNICA
DELLE MARCHE

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

Studio e sviluppo di sistemi di controllo robusto per droni

Study and development of robust control systems for drones

Candidato:
Simone Terramani

Relatore:
Prof. Gianluca Ippoliti

Correlatore:
Prof. Giuseppe Orlando

Anno Accademico 2018-2019

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

A mia nonna Lucia, esempio di vita e di resilienza

Ringraziamenti

Le prime persone che devo ringraziare per essere arrivato fin qui sono senz'altro i miei genitori, Anna Maria e Daniele che, nonostante tutto, mi hanno sostenuto lungo questo mio percorso universitario e mi hanno permesso di intraprenderlo e di portarlo a termine. Tutto questo non sarebbe stato possibile senza di loro.
Un ringraziamento va ai miei nonni Gabriele e Pina ed in particolare a nonna Lucia che, per cinque anni, con le sue telefonate quotidiane è riuscita a strapparmi un sorriso con la sua sagacia e, spesso, a tirarmi su di morale con il suo instancabile spirito.
Un ringraziamento va ai miei zii, Cinzia e Peppino, che non mi hanno fatto mai mancare il loro supporto ed affetto (ed il caffè).
Un ringraziamento va a Lucia, la mia metà. É lei che da un senso a tutti gli sforzi che ho compiuto in questi anni ed il suo amore è per me una quotidiana linfa vitale.
Un ringraziamento va agli amici di una vita, Gianmarco e Riccardo, sempre pronti ad ascoltare e a farmi sentire a casa anche quando da casa ero spesso lontano.
Un ringraziamento va a Rosamaria, mia amica e coinquilina, che mai mi ha fatto mancare il suo affetto nonostante la distanza che ci ha diviso (e che mi ha dovuto sopportare come coinquilino per ben tre anni).
Un ringraziamento va a Nicolas e ad Orlando, miei attuali coinquilini ed amici, sempre pronti a sostenermi e ad alleggerirmi le giornate con un sorriso o una battuta scherzosa.
Un ringraziamento va infine a tutte le persone che ho incontrato durante questo bellissimo e faticoso percorso che, in un modo o nell'altro, hanno contribuito a rendermi la persona che sono diventato.

Ancona, Ottobre 2019

Simone Terramani

Sommario

Nella presente tesi è stato modellato un drone avente quattro motori/rotori (quadricottero); sono stati studiati e sviluppati due controllori non lineari di tipo sliding mode (uno non robusto ed uno robusto); gli algoritmi di controllo sviluppati sono stati in un primo momento testati in simulazione e sono stati in seguito implementati sul sistema reale e le relative prove sperimentali analizzate.
Nel Capitolo 1 è presente una panoramica introduttiva sui droni, un breve excursus storico ed i vari settori (civili e non) nei quali essi trovano impiego, con un focus particolare nell'ambito della robotica educativa.
Nel Capitolo 2 viene descritto e particolarizzato il drone che è stato utilizzato all'interno della presente tesi, ovvero il *Parrot Mambo*. Sono qui descritte le specifiche tecniche come dimensioni, peso e sensoristica, corredate dalle immagini dello stesso.
Nel Capitolo 3 è presente la modellazione matematica che è stata effettuata per il drone in questione. Dopo una premessa sul significato della modellazione matematica e sugli angoli di Eulero sono introdotte le equazioni del modello cinematico e dinamico. Il capitolo si conclude con la caratterizzazione matematica di quest'ultime.
Nel Capitolo 4 è presentato l'ambiente MATLAB & Simulink utilizzato per raggiungere gli obiettivi di controllo prefissati. Le singole componenti software sono analizzate e una particolare attenzione viene dedicata allo schema di controllo e alla sua organizzazione.
Nel Capitolo 5, dopo una breve panoramica sul concetto di controllo, viene introdotto il controllo sliding mode. Successivamente questo viene specializzato e caratterizzato per il drone in esame. Il capitolo si conclude presentando le equazioni costitutive il controllore sliding mode non robusto.
Nel Capitolo 6, dopo aver spiegato il concetto di disturbo per un sistema ed il significato della robustezza del controllo, viene introdotto il controllore robusto e vengono caratterizzate le sue equazioni.
Nel Capitolo 7 vengono esposti i risultati sperimentali ottenuti dall'applicazione dei controllori di cui al capitolo 5 e 6. Vengono prima mostrati i risultati del controllore non robusto in simulazione e valutati con l'indicatore IAE (Integral Absolute Error). Successivamente vengono mostrati, sempre in simulazione, i risultati dati dall'introduzione del disturbo (instabilità del sistema) e di come questi possano essere migliorati grazie al controllore robusto. L'ultima parte è dedicata alle prove sperimentali sul drone reale, dove vengono confermati i risultati e le considerazioni fatte per le prove in simulazione.
Nel Capitolo 8 vengono presentate le conclusioni che discendono dal presente lavoro di tesi ed i possibili sviluppi futuri.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Droni e quadricotteri

1.1 Introduzione

Il lavoro della presente tesi riguarda la modellazione ed il controllo del drone *Mambo*, prodotto e commercializzato dalla compagnia francese *PARROT*. E' dunque opportuno, prima di presentare il lavoro svolto, descrivere la rilevanza che i droni come quello di cui sopra rivestono nel mercato e nella società attuale ed i vari ambiti in cui questi trovano applicazione.

1.2 Storia

Il termine "drone" è riferito a quegli aerei senza pilota a bordo, impiegati prevalentemente per uso militare ma che possono anche avere una funzione civile.

Coniato quasi 100 anni fa, il termine drone trae le sue origini dal ronzio dei primi modelli somigliante al rumore che fa il maschio dell'ape, il fuco, in inglese per l'appunto drone. Oggi questi aerei possono volare silenziosamente per oltre 20 ore su un'area assai vasta e raccogliere informazioni di ogni tipo, oltre a essere in grado di prelevare campioni di nubi vulcaniche, chimiche o nucleari [1].

Esistono diversi modelli di drone, il cui peso va da meno di un chilogrammo a diverse tonnellate, come il Global Hawk della Northrop Grumman, che può volare a 40 mila metri di altezza ed in qualsiasi condizione meteo. Nella tabella seguente, a titolo puramente esemplificativo, vengono riportare alcune delle principali tipologie di drone.

Tabella 1.1: Alcune delle possibili tipologie di drone

Acronimo	Raggio Operativo (Km)	Quota di volo (m)	Durata del volo (h)
Nano (η)	<1	100	<1
Micro (μ)	<10	250	1
Short Range (SR)	30 - 70	3000	3 - 6
Medium Range (MR)	70 - 200	5000	6 - 10
Low Altitude (LA)	>250	50 - 9000	0,5 - 1

Inoltre, non dovendo trasportare passeggeri, i droni non sono pressurizzati e possono quindi volare ad altezze precluse agli aerei di linea. Quelli che pesano alcune decine di chili sono guidati a vista da una piccola attrezzatura portatile mentre quelli di peso maggiore sono guidati via satellite da complesse stazioni di terra, che sono paragonabili a quelle degli aeromobili di linea.

1.2.1 UAV

Dal momento in cui hanno cominciato a svilupparsi gli impieghi civili di queste macchine, si è diffuso il termine *unmanned air vehicle* (UAV) ovvero veicolo senza pilota (in realtà, gli unmanned vehicles esistono da tempo: basti pensare a certi treni delle metropolitane o a certi mezzi sottomarini). Da un punto di vista formale e più generico (estendendo quindi il discorso anche a veicoli terrestri e sottomarini), gli *Unmanned vehicles* possono essere definiti come veicoli in grado di portare a termine un compito senza l'ausilio dell'uomo. Per poter riuscire in tali compiti, i veicoli autonomi fanno affidamento su:

- **sensori**: forniscono l'informazione relativa all'ambiente esterno o agli stati interni del sistema;
- **attuatori**: realizzano fisicamente il movimento desiderato;
- **controllori**: pilotano gli attuatori in modo da portare a termine il compito richiesto.

Chiaramente anche il drone *Parrot Mambo*, oggetto della presente tesi, è caratterizzato da questi elementi funzionali che verranno analizzati con maggior dettagli nel capitolo successivo.

Droni di dimensioni inferiori possono essere impiegati per la sorveglianza di porti, aeroporti, centrali nucleari o altri siti sensibili per prevenire atti terroristici. Questi aerei sono poi impiegati per la raccolta di campioni da nubi tossiche, interventi a difesa della popolazione, la lotta agli incendi o la prevenzione (early warning system) di disastri naturali (soprattutto inondazioni) che nella sola Europa procurano ogni anno danni per più di 300 miliardi di euro, oltre alla perdita di vite umane. Durante il disastro nucleare di Fukushima in Giappone, i droni sono stati usati per raccogliere campioni della nube nucleare nelle zone circostanti e guidare l'evacuazione della popolazione.

1.3 Impieghi

Il successo e la diffusione capillare che questi oggetti stanno avendo nella società odierna è dovuta al fatto che i droni possono essere impiegati in una gamma eterogenea di compiti e funzioni. Di seguito, a scopo esemplificativo, ne verranno riportati alcuni [2]:

- **ricerca e soccorso**: nelle operazioni di ricerca e soccorso, ogni secondo può valere moltissimo. Per funzionare nel modo più efficiente possibile, è importante avere una rapida panoramica della situazione, ovvero un punto di vista dal cielo. Si può monitorare qualsiasi disastro ambientale avvalendosi dei droni: mentre gli aerei e gli elicotteri richiedono un lasso di tempo per essere pronti (causa presenza umamna a bordo), un drone UAV può entrare in azione immediatamente, senza alcuna perdita di tempo. Le loro protezioni sono in grado di resistere alla pioggia o alla neve, oppure a temperature estreme come in caso di incendi. Inoltre, con l'aggiunta di opportuni sensori, possono essere misurate in tempo reale delle specifiche condizioni di interesse (temperature, presenze di particolari gas o sostanze nell'aria et cetera);
- **ispezioni e sopralluogo**: per garantire una fornitura dell'energia elettrica, del gas e del petrolio, le applicazioni dei droni si estendo anche nel campo delle ispezioni. Grazie alla loro dimensione e struttura, i quadricotteri dotati di sistemi GPS, posso essere utilizzati per ispezionare un'area in maniera del tutto autonoma, e ritornare al punto del decollo, senza il bisogno di istruzioni e/o navigazione del pilota.

- **sicurezza e sorveglianza**: la sicurezza civile è un argomento di grande importanza. Per proteggere i cittadini da minacce di vario tipo (monitoraggio costiero, sorveglianza attività illegali,protezione da scavi illegali, operazioni anti bracconaggio, monitoraggio inondazioni, controllo dei rifiuti, attacchi terroristici), ci si può avvalere dei droni. Gli elicotteri sono una fonte importante, ma costosi e necessitano di tempo prima di essere resi operativi. L'applicazione dei droni in questo campo è più veloce e conveniente. In alcuni casi, possono essere muniti di uno scanner laser per registrare un caso di sinistro stradale, elaborando così le dinamiche dell'incidente.

- **scienza e ricerca**: nei progetti scientifici e di ricerca, avere una visuale aerea può rivelarsi vantaggioso in quanto permette di osservare una scena da diversi punti di vista e angolazioni; spesso inoltre, non bisogna disturbare o contaminare l'ambiente in cui si opera. I droni vengono quindi utilizzati per attività come: ricerche archeologiche; contaminazione zone nucleari; sorveglianza ghiacciai; mappatura siti di scavo; fotogrammetria; sorveglianza dei mammiferi marini.

Figura 1.1: Drone per scopi di sicurezza

(a) Drone per la sorveglianza urbana.

(b) Drone per il sopralluogo scientifico

Figura 1.2: I vari ambiti di utilizzo dei droni

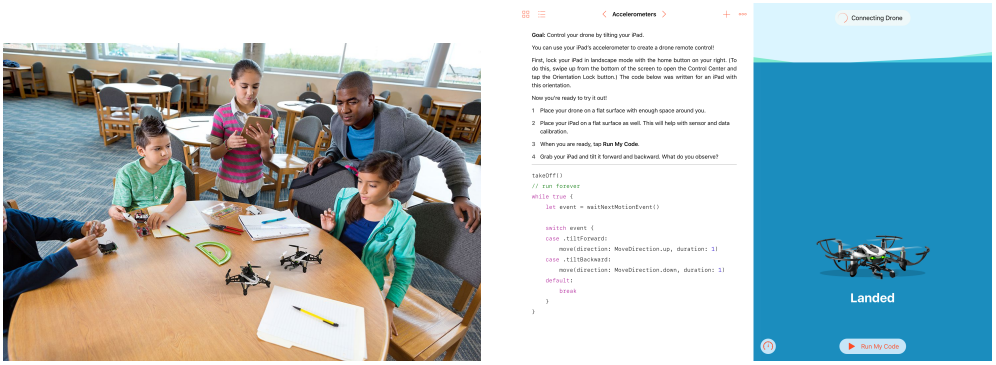
1.3.1 Robotica educativa

La diffusione capillare dei droni ne ha comportato anche la riduzione dei costi e ne ha facilitato quindi una ulteriore penetrazione nei vari settori di impiego: quello che si vuole analizzare nel dettaglio in questo paragrafo è il settore della robotica educativa. Il (mini)drone *Parrot Mambo* è un esempio di un nuovo modo di portare l'insegnamento delle discipline *STEM* (Scienza, Tecnologia, Ingegneria (Engineering), Matematica) nelle classi. Offre una varietà di strumenti e approcci che lo rendono adatto agli studenti dalle scuole elementari

alle università e fino ai corsi di dottorato di ricerca: supporta infatti le migliori piattaforme di *coding* che permettono all'utente finale di avere differenzi opzioni per affrontare problemi eterogenei riguardanti per l'appunto le materie STEM. In particolare [3]:

- per le scuole primarie e secondarie di primo grado offre la codifica di blocchi con piattaforme come *Tynker*, *Blocky* ed *Apple Swift Playground*.
- per le scuole secondarie di secondo grado e le università offre compatibilità con linguaggi di programmazione come *JavaScript* e *Python* e compatibilità con le piattaforme di *Mathworks MATLAB & Simulink*.

L'ambiente *MATLAB & Simulink* di *Mathworks* è quello che è stato utilizzato per svolgere il lavoro di implementazione di un algoritmo di controllo non lineare robusto (controllo *sliding mode*) al fine di dimostrare la flessibilità e la versatilità di questo nuovo ecosistema educativo: dalla programmazione a blocchi (tramite *Simulink*) per la prototipazione, all'implementazione ed al collaudo di approcci di controllo sperimentali all'avanguardia.



(a) Esempio di classe STEM (b) Interfaccia di Apple Swift Playground

Figura 1.3: Come il *Parrot Mambo* è utilizzato in ambienti di robotica educativa

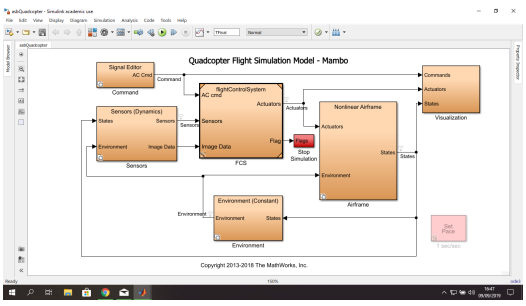


Figura 1.4: Interfaccia grafica dell'ambiente *MATLAB & Simulink* utilizzato

Capitolo 2

Parrot Mambo

2.1 Caratteristiche tecniche

Come già anticipato, il drone (spesso chiamato anche minidrone date le sue dimensioni molto contenute) che è stato utilizzato in questa tesi è il *Mambo*, prodotto dall'azienda francese *Parrot*. Di seguito ne verranno mostrate le principali caratteristiche tecniche:

- **Dimensioni**: 18×18 cm
- **Peso**:
  - Peso: 63 g (senza paraurti o accessori)
  - Peso con telecamera: 73 g
- **Energia**:
  - Batteria LiPo 660 mAh
  - 8 min di autonomia con accessori o paraurti collegati
  - 10 min di autonomia senza accessori o paraurti
  - 30 minuti di ricarica con un caricatore da 2,1 Ampere
- **Sensoristica per la stabilizzazione**:
  - IMU (Unità di misura inerziale) composta da un accelerometro a 3 assi e da un giroscopio a 3 assi
  - Sensore di altitudine ad ultrasuoni (misura fino a 4 m di altitudine)
  - Sensore di pressione
  - Telecamera verticale da 60 FPS (Frames per Second) che permette la misurazione della velocità

2.1.1 IMU: Inertial Measurement Unit

La creazione di un sistema sensoriale in grado di percepire il movimento di un oggetto rispetto all'ambiente circostante è un problema rilevante nella robotica mobile: gli algoritmi di controllo ,infatti, sono progettati si tali dati sensoriali, pertanto la loro disponibilità ed affidabilità è un requisito fondamentale. Un sistema di navigazione inerziale (*Inertial Navigation System, INS*) è un sistema in grado di acquisire l'informazione di navigazione dell'oggetto (posizioni e velocità lineari ed angolari) nel suo spazio di riferimento e di trasformarla nello spazio di riferimento "assoluto". In molte applicazioni però, come nel caso di questo lavoro, non è necessario conoscere l'informazione di navigazione completa, ma è sufficiente conoscere le sole misure inerziali (accelerazione lineare e velocità angolare): in questi casi si parla di unità di misura inerziali (*Inertial Measurement Unit, IMU*) [4].

2.1.2 Sensore ad ultrasuoni (e sensore di pressione)

I sistemi SONAR (*sound navigation and ranging*) sono impiegati in robotica mobile come soluzione a costo ridotto per la misura della distanza tra il robot e l'ambiente circostante (nel caso della presente tesi per rilevare la distanza tra il drone ed il terreno). Il loro principio di funzionamento si basa sul calcolo del tempo di volo impiegato da un'onda ultrasonora per percorrere la distanza da e per il sensore: essi sono al tempo stesso generatori di segnale e trasduttori. Il sensore di pressione è di ausilio al sensore ad ultrasuoni per il calcolo dell'altitudine del drone.

2.2 Immagini descrittive del drone

In questa sezione verranno mostrate alcune immagini del *Parrot Mambo*, che serviranno a metterne in evidenza le cartteristiche salienti.

(a) Visione frontale del *Parrot Mambo* (b) Vione del retro del *Parrot Mambo*

Figura 2.1: Il fronte e il retro del *Parrot Mambo*

Nella figura 2.1 si possono vedere la parte frontale e superiore del *Parrot Mambo*. Nella figura 2.1a vi sono i due LED frontali che cambiano colore a seconda dello stato del drone:

- colore verde: il drone è pronto al volo;
- colore rosso: la batteria del drone è scarica;
- colore arancione: il firmware del drone è in fase di scrittura (questo punto verrà approfondito in (4)).

Nella figura 2.1b si possono notare l'alloggiamento che ospita la batteria che alimenta il drone e, in basso a destra la porta *micro-usb* per la ricarica ed il collegamento al computer.

(a) Visione dall'alto del *Parrot Mambo* (b) Visione dal basso del *Parrot Mambo*

Figura 2.2: Il fronte e il retro del *Parrot Mambo*

Nella figura 2.2 sono mostrate la parte superiore ed inferiore del *Parrot Mambo*. Nella figura 2.2a è presente l'alloggiamento per eventuali accessori esterni al drone (telecamera FPV-first person view, pinze) e si possono inoltre notare che le eliche del drone sono di due colori differenti: quelle bianche sono quelle anteriori mentre quelle nere sono le posteriori.

Nella figura 2.2b è presente una porzione della sensoristica. In ordine dall'alto al basso:

- sensore di pressione: sporgenza nera in corrispondenza della punta del drone;
- sensore ad ultrasuoni: cerchio metallico;
- telecamera 60 FPS: piccolo foro immediatamente sotto al sensore ad ultrasuoni

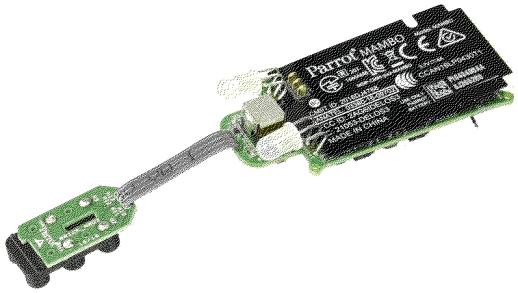
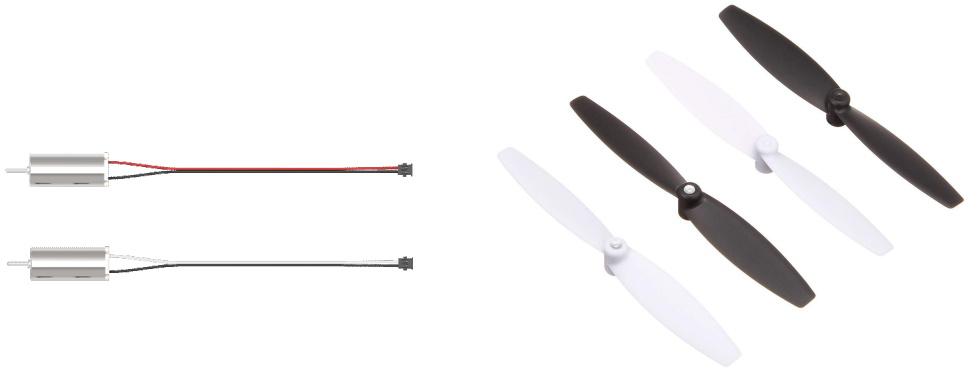


Figura 2.3: Printed circuit board (PCB) del *Parrot Mambo*

Nella figura 2.3 è mostrata la circuiteria del *Parrot Mambo* dove sono alloggiati i circuiti per l'alimentazione, il pilotaggio degli attuatori, la logica di controllo e la IMU (giroscopio a 3 assi ed accelerometro a 3 assi).



(a) Motori "A" e "C" del *Parrot Mambo*

(b) Eliche "A" e "C" del *Parrot Mambo*

Figura 2.4: Gli attuatori del *Parrot Mambo*

A concludere questa breve panoramica sulla componentistica del *Parrot Mambo* vengono mostrati i motori e le rispettive eliche 2.4.
I due motori, presenti in figura 2.4a sono il motore A, che sta per "Anti-Clock" e che gira in senso antiorario ed il motore C, che sta per "Clock" e che gira in senso orario. Entrambi sono di tipo *coreless* e dimensioni di $8,5 \times 20$ mm:

- il motore "A" si posiziona sulle parti anteriore destra e posteriore sinistra;
- il motore "C" si posiziona sulle parti anteriore sinistra e posteriore destra.

In modo del tutto analogo vi sono due tipi di eliche, identificabili grazie alla dicitura "C" (senso orario) e "A" (senso antiorario):

- anteriore sinistro e posteriore destro: "C"
- anteriore destro e posteriore sinistro: "A"

Queste eliche in plastica sono state studiate per ridurre al minimo il consumo elettrico dei motori e, al tempo stesso, assicurare una spinta massima ai motori.

A chiusura di questa sezione viene mostrata una tabella con tutti i parametri che sono necessari per l'implementazione dei controllori nelle sezioni 5 e 6:

Tabella 2.1: Parametri del Parrot Mambo

Parametro	Valore	Unità di misura
m	0,063	Kg
J_{xx}	5,82857e-05	$Kg.m^2$
J_{yy}	7,16914e-05	$Kg.m^2$
J_{zz}	1,0e-04	$Kg.m^2$
d	0,0624	m
J_{tp}	4,08375e-07	$Kg.m^2$
C_T	0,0107	$N.s^2$
C_q	7,826375278505369e-04	$N.m.s^2$
g	9,81	m/s^2

Dove:

- m è la massa del drone;
- J_{ii} rappresenta il momento principale di inerzia lungo l'asse $i - esimo$;
- d è la lunghezza del corpo del drone;
- J_{tp} è il momento risultante rotazionale di inerzia attorno all'asse del motore;
- C_T è il coefficiente di spinta che serve a trasformare la velocità dell'$i - esimo$ rotore espressa in radianti al secondo nella forza F_i;
- C_q è il coefficiente di coppia che serve a trasformare la velocità dell'$i - esimo$ rotore espressa in radianti al secondo nel momento M_i;
- g è la costante di gravitazione universale

Capitolo 3

Modellazione del Parrot Mambo

3.1 Modello matematico

Dopo aver presentato, nel capitolo 2, le specifiche tecniche del *Parrot Mambo*, ne verranno ora ricavate le caratteristiche essenziali, ovvero il modello matematico.

3.1.1 La modellazione matematica

> "Le scienze non cercano di spiegare, a malapena tentano di interpretare, ma fanno soprattutto dei modelli. Per modello s'intende un costrutto matematico che, con l'aggiunta di certe interpretazioni verbali, descrive dei fenomeni osservati. La giustificazione di un siffatto costrutto matematico è soltanto e precisamente che ci si aspetta che funzioni - cioè descriva correttamente i fenomeni in un'area ragionevolmente ampia. Inoltre esso deve soddisfare certi criteri estetici - cioè, in relazione con la quantità di descrizione che fornisce, deve essere piuttosto semplice." (John von Neumann)

In linea generale, si è interessati a descrivere l'evoluzione nel tempo di oggetti che identifichiamo rispetto a un ambiente circostante e che vengono chiamati sistemi. Quando si riescono a identificare delle ricorrenze e a descriverle in un linguaggio matematico, allora si ha un modello matematico. Quest'ultimo, dunque, è la rappresentazione di un fenomeno: tale rappresentazione non è discorsiva o a parole, ma formale, espressa cioè in linguaggio matematico. Costruendo un modello, bisogna necessariamente trascurare alcuni aspetti ritenuti secondari e concentrarsi su una parte di un grande quadro, su un particolare insieme di caratteristiche del sistema che si vogliono descrivere. Da ciò segue che:

- I modelli, tutti e necessariamente, sono astrazioni della realtà;
- I modelli non possono descrivere completamente una realtà troppo complicata per poter essere descritta in ogni suo aspetto.

I modelli matematici quindi possono essere visti come una rappresentazione approssimata della realtà, espressa attraverso una relazione in termini logico-matematici tra le variabili caratteristiche del sistema. Usando un sistema di calcolo è inoltre possibile utilizzare il modello al fine di ottenere un comportamento il più possibile vicino all'andamento del sistema reale, per investigare il suo comportamento futuro e sviluppare metodi di previsione, controllo e diagnosi. A concludere, si può affermare che esistono fondamentalmente 3 macro-tipologie di identificazione:

- **black box**: utile per avere conoscenza di uno o più parametri del processo quando di esso non è disponibile alcun modello matematico;

- **grey box**: utile per stimare parametri di un modello noto e di migliorare la conoscenza dello stesso;
- **white box**: utile quando si conoscono le dinamiche fisiche in gioco ed è possibile codificarle tramite equazioni matematiche

Il modello, nel caso in esame, è stato ricavato tramite identificazione di tipo *white box*.

3.1.2 Premessa: angoli di Eulero

L'orientazione di un veicolo è usualmente individuata dagli angoli di rollio, beccheggio ed imbardata, che sono utilizzati in aeronautica e che spesso sono chiamati *angoli RPY*, dall'ingese Roll, Pitch and Yaw (si veda la figura). Si consideri un sistema di riferimento solidale sol corpo rigido (un velivolo, nel caso in esame il *Parrot Mambo*), con l'asse x diretto longitudinalmente, l'asse y diretta trasversalmente e l'asse z ortogonale al piano del velivolo. Una rotazione di un angolo ϕ attorno all'asse x è detta *rollio* (in inglese roll); una rotazione di un angolo θ intorno all'asse y detta *beccheggio* (in inglese pitch) ed una rotazione di un angolo ψ intorno all'asse z detta *imbardata* (in inglese yaw). Si eseguono le rotazioni di tre angoli ψ, θ e ϕ rispettivamente intorno al terzo, al secondo ed al primo asse coordinato, che si possono riassumere nello schema seguente:

$$(\mathbf{i},\mathbf{j},\mathbf{k}) \rightarrow (\mathbf{i}^{'},\mathbf{j}^{'},\mathbf{k}^{'}) \rightarrow (\mathbf{i}^{''},\mathbf{j}^{''},\mathbf{k}^{''}) \rightarrow (\mathbf{I},\mathbf{J},\mathbf{K})$$

dove **I**,**J**,**K** sono i nuovi versori al termine delle tre rotazioni.

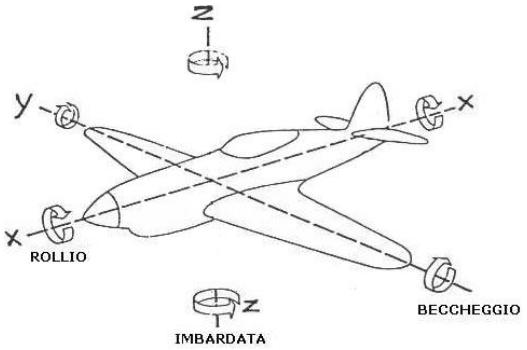


Figura 3.1: Angoli *RPY*

3.1.3 Considerazioni generali

Per comprendere il principio di funzionamento di un drone, nel caso in esame un quadricottero (quattro motori), è utile fare dei paragoni con un elicottero dotato di due rotori, uno principale e uno posteriore. La grande stabilità di un drone è dovuta alla geometria che si utilizza in fase di costruzione disponendo i rotori in una configurazione a croce [5]. Come mostrato nella figura 3.2 , due rotori girano in direzione opposta rispetto agli altri due. In particolare, quando tutti i rotori ruotano con la stessa velocità angolare, con i rotori 1 e 3 rotanti in senso orario ed i rotori 2 e 4 in senso antiorario, l'accelerazione angolare attorno all'asse di imbardata è nulla (non vi è una rotazione del drone) e dunque il rotore posteriore di un classico elicottero (il quale ha lo scopo di non permettere la rotazione) si rende non necessario. Se si vuole l'effetto contrario, cioè generare un angolo di yaw, si deve creare una

discrepanza di velocità tra i rotori che girano in senso orario rispetto a quelli che ruotano in senso anti-orario. Per quanto riguarda lo spostamento sull'asse verticale occorre impostare una velocità di rotazione uniforme tra i quattro motori. Se invece si volesse procedere con una traslazione sul piano si deve procedere variando le velocità delle eliche non opposte ma adiacenti. Così facendo si genera un momento torcente che inclinerà il velivolo (Pitch / Roll) con la conseguente traslazione. Le figure in 3.3 illustrano che tipologia di spinta è necessaria per la movimentazione del drone.

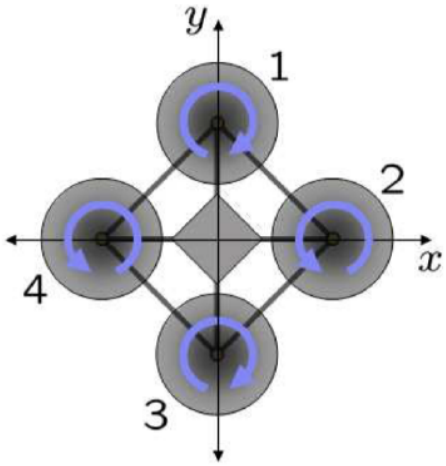
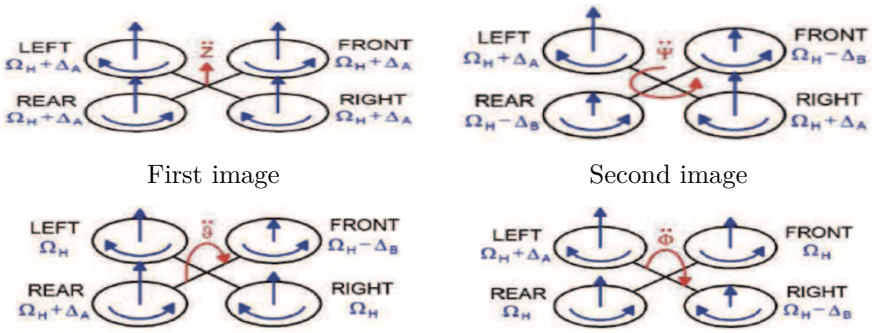


Figura 3.2: Schema geometrico di un quadricottero



First image Second image

Fourth image Fifth image

Figura 3.3: Movimentazione quadricottero

3.1.4 Modello cinematico

Il drone è un sistema sottoattuato. Possiede 6 gradi di libertà (*DOF: degree of freedom*) che sono, come mostrato nella figura 3.4, le tre coordinate cartesiane X,Y,Z ed i tre angoli di assetto ϕ, θ, ψ, come già mostrato in 3.1, che sono rispettivamnet gli angoli di roll, di pitch e di yaw. D'altro canto è equipaggiato solamente con quattro motori (rotori): è quindi impossibile raggiungere direttamente uno stato desiderato di 6 *DOF*, ma di 4 al massimo. In

ogni caso, grazie alla sua struttura, è facile scegliere 4 controlli (variabili di controllo) che sono in grado di imporre l'altitudine e l'assetto (posizione angolare nello spazio) desiderato.

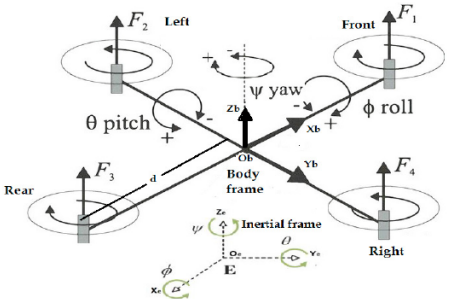


Figura 3.4: Struttura del drone e cinematica

Questi controlli sono i seguenti:

- *Spinta* U_z: una forza verticale che si ottiene aumentando o diminuendo la velocità dei motori della stessa quantità;
- *Roll* U_ϕ: rappresenta una coppia rispetto all'asse x del *body frame* (sistema di riferimento centrato nel centro di massa del drone) che si ottiene aumentando o diminuendo la velocità o dei motori di destra o di quelli di sinistra;
- *Pitch* U_θ: rappresenta una coppia rispetto all'asse y e si ottiene aumentando o diminuendo la velocità o dei motori anteriori o di quelli posteriori;
- *Yaw* U_ψ: rappresenta una coppia rispetto all'asse z e si ottiene aumentando (o diminuendo) la velocità della coppia di motori anteriori/posteriori e diminuendo (o aumentando) la velocità dei motori della coppia destra/sinistra.

Vengono utilizzati due sistemi di coordinate per descrivere il moto nello spazio del drone: il body frame inerziale (sistema di riferimento corpo) ed il sistema di riferimento fisso o assoluto. Il sistema di riferimento assoluto $\begin{pmatrix} O_e & x_e & y_e & z_e \end{pmatrix}$ è utilizzato per definire il vettore posizione del drone $\xi = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ espresso in metri ed il vettore degli angoli di Eulero $\sigma = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ espresso in radianti. Il body frame $\begin{pmatrix} O_b & x_b & y_b & z_b \end{pmatrix}$ definisce il vettore delle velocità lineari $\mu = \begin{bmatrix} u & v & w \end{bmatrix}^T$ espresso in metri al secondo, il vettore delle velocità angolari $\omega = \begin{bmatrix} P & Q & R \end{bmatrix}^T$ espresso in radianti al secondo, le forze F espresse in newton e le coppie τ espresse in newton metri. La posizione lineare del drone è descritta dalle coordinate del vettore posizione che punta dall'origine del sistema di riferimento assoluto all'origine del body frame. L'assetto (o posizione angolare) del drone è rappresentato dalle coordinate del vettore degli angoli di Eulero, che fornisce l'orientazione della terna corpo rispetto alla terna mondo (o assoluta). Il passaggio da un sistema di riferimento all'altro si ottiene tramite tre rotazioni consecutive attorno gli assi x, y e z che mappano la terna mondo con la terna corpo. Ipotizzando che l'ordine delle rotazioni sia prima roll, poi pitch ed infine yaw, si ottiene la seguente matrice di rotazione:

$$R_0 = \begin{bmatrix} C_\psi C_\theta & -S_\psi C_\theta + C_\psi S_\theta S_\phi & S_\psi S_\phi + C_\psi S_\theta C_\phi \\ S_\psi C_\theta & C_\psi C_\phi + S_\psi S_\theta S_\phi & C_\psi S_\phi + S_\psi S_\theta C_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \tag{3.1}$$

dove C, S e T rappresentano rispettivamente le funzioni coseno, seno e tangente.

Sia ora $\dot{\xi} = \begin{bmatrix} \dot{X} & \dot{Y} & \dot{Z} \end{bmatrix}^T$ il vettore delle velocità lineari del drone rispetto alla terna mondo. La relazione tra il vettore delle velocità lineari $\dot{\xi}$ rispetto alla terna mondo ed il vettore delle velocità lineari $\mu = \begin{bmatrix} u & v & w \end{bmatrix}^T$ rispetto alla terna corpo è dato da:

$$\dot{\xi} = R_0 \mu \tag{3.2}$$

Così come si è appena fatto per le velocità lineari, è possibile trovare una relazione tra il vettore delle velocità angolari $\omega = \begin{bmatrix} P & Q & R \end{bmatrix}^T$ rispetto alla terna corpo ed il vettore delle velocità angolari $\dot{\sigma} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$ rispetto alla terna mondo. Per fare ciò, viene utilizzata una mtrice di trasformazione T_0.

$$T_0 = \begin{bmatrix} 1 & -S_\phi T_\theta & T_\theta C_\phi \\ 0 & C_\phi & -S_\phi \\ 0 & \frac{S_\phi}{C_\theta} & \frac{C_\phi}{C_\theta} \end{bmatrix} \tag{3.3}$$

Segue quindi che la relazione è:

$$\dot{\sigma} = T_0 \omega \tag{3.4}$$

3.1.5 Modello dinamico

Il modello dinamico è ricavato usando il formalismo di Newton-Eulero. La dinamica del corpo rigido a 6 gradi di libertà, sotto l'azione di forze e coppie applicate nel suo centro di massa ed espresse rispetto alla terna corpo usando il metodo di Newton-Eulero è [6]:

$$\begin{bmatrix} mI_{3,3} & 0 \\ 0 & J \end{bmatrix} \begin{bmatrix} \dot{\mu} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega \times m\omega \\ \omega \times J\omega \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \tag{3.5}$$

dove $I_{3,3}$ è un amatrice identità 3×3, $\dot{\mu}$ e $\dot{\omega}$ sono rispettivamente i vettori delle accelerazioni lineari ed angolari, F e τ sono le coppie e forze esterne, J è il momento di inerzia ed m è la massa del drone.

Per poter sviluppare correttamente l'equazione 3.5 bisogna determinare le differenti forze e coppie che agiscono sul drone. Trascurando l'effetto di trascinamento (drag effect), l'effetto hub ed il vento, le principali forze e coppie che agiscono sul sistema sono le seguenti:

- *Forze e coppie in input (ingresso)*: queste forze e coppie sono generate direttamente dai quattro motori. Come risultato della rotazione, ciascun motore mostra un momento aerodinamico M_i ed una forza di sollevamento F_i date da [7]:

$$\begin{aligned} F_i &= C_T \Omega_i^2 \\ M_i &= C_q \Omega_i^2 \end{aligned} \tag{3.6}$$

  dove C_T e C_q sono rispettivamente i coefficienti di spinta e di coppia ed Ω_i è la velocità dell'i-esimo rotore. Per esprimere gli effetti totali delle forze e delle coppie su ciascun asse, viene utilizzata la regola della mano destra rispetto alla terna corpo:

$$\begin{aligned} M_x &= F_4 d - F_2 d = C_T d(\Omega_4^2 - \Omega_2^2) \\ M_y &= F_3 d - F_1 d = C_T d(\Omega_3^2 - \Omega_1^2) \\ M_z &= -C_q(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{aligned} \tag{3.7}$$

Il vettore risultante di coppie in ingresso è quindi:

$$\tau_{in} = \begin{bmatrix} U_\phi \\ U_\theta \\ U_\psi \end{bmatrix} = \begin{bmatrix} C_T d(\Omega_4^2 - \Omega_2^2) \\ C_T d(\Omega_3^2 - \Omega_1^2) \\ C_q(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{3.8}$$

Il vettore risultante delle forze in ingresso rispetto al body frame è dato da:

$$F_{in}^B = \begin{bmatrix} 0 \\ 0 \\ U_Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ C_T(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{3.9}$$

Poichè interessa esprimere le grandezze in 3.9 rispetto alla terna mondo, è necessario premoltiplicarle per la matrice di rotazione R_0:

$$F_{in} = R_0 F_{in}^B \tag{3.10}$$

dove in 3.10 l'apice B indica che la grandezza è espressa rispetto al body frame.

- *Effetto giroscopico*: questo effetto è generato dalla rotazione dei motori ed è dato da [8]:

$$\tau_g = \begin{bmatrix} -\sum_{i=1}^{4} J_{tp} \left[\omega \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] - (-1)^i \Omega_i \end{bmatrix} \tag{3.11}$$

$$= \begin{bmatrix} -Q\Omega J_{tp} \\ P\Omega J_{tp} \\ 0 \end{bmatrix} \tag{3.12}$$

dove J_{tp} è il momento angolare di inerzia totale attorno l'asse del motore e Ω è la somma ($\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$) delle velocità dei 4 motori espresse in radianti al secondo.

- *Forza di gravità*: questo effetto può essere espresso direttamente nella terna mondo. Il vettore che rappresenta la gravità è:

$$F_G = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{3.13}$$

dove g è la costante di accelerazione gravitazionale.

Si hanno ora tutti gli elementi per sviluppare correttamente l'espressione 3.5. Si può quindi esprimere completamente la dinamica completa del drone a 6 gradi di libertà, con il sottosistema traslazionale espresso rispetto alla terna mondo ed il sottosistema rotazionale

rispetto alla terna corpo. Le equazioni sono le seguenti:

$$\begin{cases} \ddot{X} = (S_\phi S_\psi + C_\psi S_\theta C_\phi)\frac{U_z}{m} \\ \ddot{Y} = -(S_\phi C_\psi + S_\psi S_\theta C_\phi)\frac{U_z}{m} \\ \ddot{Z} = -g + C_\theta C_\phi \frac{U_z}{m} \\ \dot{P} = \frac{1}{J_{xx}}(U_\phi + (J_{yy} - J_{zz})QR + Q\Omega J_{tp}) \\ \dot{Q} = \frac{1}{J_{yy}}(U_\theta + (J_{zz} - J_{xx})PR - P\Omega J_{tp}) \\ \dot{R} = \frac{1}{J_{zz}}(U_\psi + (J_{xx} - J_{yy})PQ) \end{cases} \tag{3.14}$$

E' inoltre più conveniente scrivere il sistema rotazionale rispetto alla terna mondo in quanto il lavoro della presente tesi è quello di studiare e sviluppare algoritmi di controllo per gli angoli di roll, pitch e yaw che sono espressi rispetto alla terna mondo. A tal fine, si introducono le seguenti ipotesi semplificative:

- il drone si suppone che lavori in condizioni di hovering (o lineari): è una particolare tipologia di volo che si verifica quando il drone staziona in aria a velocità nulla e quota costante. In questo caso si parla, appunto, di volo stazionario o volo puntiforme, in quanto la rotta del drone nello spazio indica idealmente un punto;

- si considerano che i possibili angoli di roll e pitch siano "piccoli", ovvero minori di 20 gradi.

In queste condizioni, segue che la matrice di trasformazione presente in 3.3 sia approssimabile con una matrice identità:

$$T_0 = \begin{bmatrix} 1 & -S_\phi T_\theta & T_\theta C_\phi \\ 0 & C_\phi & -S_\phi \\ 0 & \frac{S_\phi}{C_\theta} & \frac{C_\phi}{C_\theta} \end{bmatrix} \simeq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

Da queste assunzioni e da quanto detto fin'ora segue che la dinamica del drone è completamente descritta da:

$$\begin{cases} \ddot{X} = (S_\phi S_\psi + C_\psi S_\theta C_\phi)\frac{U_z}{m} \\ \ddot{Y} = -(S_\phi C_\psi + S_\psi S_\theta C_\phi)\frac{U_z}{m} \\ \ddot{Z} = -g + C_\theta C_\phi \frac{U_z}{m} \\ \ddot{\phi} = \frac{1}{J_{xx}}(U_\phi + (J_{yy} - J_{zz})\dot{\theta}\dot{\psi} + \dot{\theta}\Omega J_{tp}) \\ \ddot{\theta} = \frac{1}{J_{yy}}(U_\theta + (J_{zz} - J_{xx})\dot{\phi}\dot{\psi} - \dot{\phi}\Omega J_{tp}) \\ \ddot{\psi} = \frac{1}{J_{zz}}(U_\psi + (J_{xx} - J_{yy})\dot{\phi}\dot{\theta}) \end{cases} \tag{3.16}$$

Capitolo 4

Ambiente MATLAB & SIMULINK

4.1 Considerazioni generali

MATLAB (abbreviazione di Matrix Laboratory), software prodotto da *Mathworks*, è un ambiente per il calcolo numerico e l'analisi statistica [9]. Consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e interfacciarsi con altri programmi. MATLAB è usato da milioni di persone nell'industria e nelle università per via dei suoi numerosi strumenti a supporto dei più disparati campi di studio applicati e funziona su diversi sistemi operativi, tra cui Windows, Mac OS, e Linux (Ubuntu) [10]. Simulink è un ambiente a blocchi per modellare, simulare e analizzare sistemi dinamici come quelli di controllo e quelli per l'elaborazione dei segnali ed è strettamente integrato con MATLAB. Le caratteristiche principali dell'ambiente MATLAB sono le seguenti [11]:

- combina un ambiente desktop ottimizzato per l'analisi iterativa e i processi di progettazione con un linguaggio di programmazione che esprime le operazioni matematiche con matrici e array in modo diretto;
- dispone di toolbox sviluppati professionalmente, rigorosamente testati e interamente documentati;
- offre la possibilità di eseguire analisi di dati su cluster, GPU e cloud solo con lievi modifiche al codice;
- il codice generato è pronto per la produzione, così da poter accedere direttamente ai sistemi cloud e aziendali, integrandosi con sistemi business e fonti di dati;
- converte automaticamente algoritmi MATLAB in codice C/C++ e HDL per essere implementati in dispositivi embedded;
- lavora con Simulink per supportare la progettazione model-based, che è utilizzata per la simulazione multidominio, la generazione automatica del codice e i test e le verifiche dei sistemi embedded.

Sono proprio le ultime due features quelle che sono state sfruttate per il presente lavoro di tesi: grazie ad una struttura software preesistente si è potuto, in un primo momento, prendere confidenza con lo schema di controllo ed in seguito sfruttarla per poter raggiungere gli obiettivi e gli scopi desiderati.

In particolare, sono stati utilizzati i seguenti strumenti software di Matlab & Simulink:

- **Simulink Support Package for Parrot Minidrones**: permette di modificare il firmware preesistente del Parrot Mambo, istallandone uno che ne consente l'interfacciamento con l'ambiente MATLAB & Simulink;

- **Aerospace Toolbox**: fornisce strumenti e funzioni per analizzare la navigazione e l'ambiente di veicoli aerospaziali e visualizzare il loro volo utilizzando la normale strumentazione di bordo presente in un cockpit o un simulatore di volo. Inoltre, consente di rappresentare l'aerodinamica del veicolo e includere modelli di ambiente convalidati per atmosfera, gravità, vento, altitudine e campo magnetico;

- **Aerospace Blockset**: fornisce blocchi Simulink per la modellazione, la simulazione e l'analisi di veicoli aerospaziali. È possibile incorporare dinamiche del veicolo, modelli convalidati dell'ambiente di volo e del comportamento del pilota, quindi collegare il modello al simulatore di volo per visualizzare i risultati della simulazione. Grazie alle operazioni integrate di matematica applicate all'ambito aerospaziale, al sistema di coordinate e alle trasformazioni spaziali è possibile descrivere il comportamento dei corpi a tre e sei gradi di libertà come, ad esempio, il Parrot Mambo;

- **Simulink *3D* Animation**: fornisce app per collegare i modelli Simulink e gli algoritmi MATLAB agli oggetti grafici 3D. Consente di modificare la posizione, la rotazione, la scala e altre proprietà degli oggetti durante la simulazione desktop o in tempo reale. Simulink 3D Animation include inoltre editor e visualizzatori per il rendering e l'interazione con scene virtuali;

- **Embedded Coder**: genera codice C e C ++ leggibile, compatto e veloce per processori embedded. Estende MATLAB Coder e Simulink Coder con ottimizzazioni avanzate per un controllo preciso delle funzioni, dei file e dei dati generati. Permette di incorporare strumenti di terze parti per la creazione di eseguibili e fornisce report di tracciabilità, documentazione del codice e verifica software automatizzata. Embedded Coder, infine, offre pacchetti di supporto con ottimizzazioni avanzate e driver di dispositivo per hardware specifico (come il Parrot Mambo).

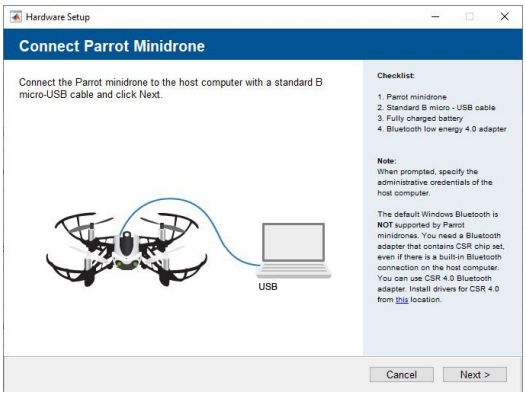


Figura 4.1: Interfaccia del Simulink Support Package for Parrot Minidrones

Riassumendo si può quindi affermare che il *Simulink Support Package* serve a far comunicare l'ambiente MATLAB & Simulink con il Parrot Mambo, l' *Aerospace ToolBox* e l' *Aerospace Blockset* servono per l'implementazione del modello e della sua virtualizzazione, *Simulink 3D Animation* serve per la visualizzazione al calcolatore del drone e delle simulazioni mentre l' *Embedded Coder* trasforma il codice sviluppato su MATLAB e Simulink in codice C/C++ eseguibile dal drone.

4.2 asbQuadcopter

Verrà ora descritta in maggiore dettaglio la struttura software (preesistente) che è stata modificata e sfruttata per raggiungere gli obiettivi di controllo desiderati. Una prima

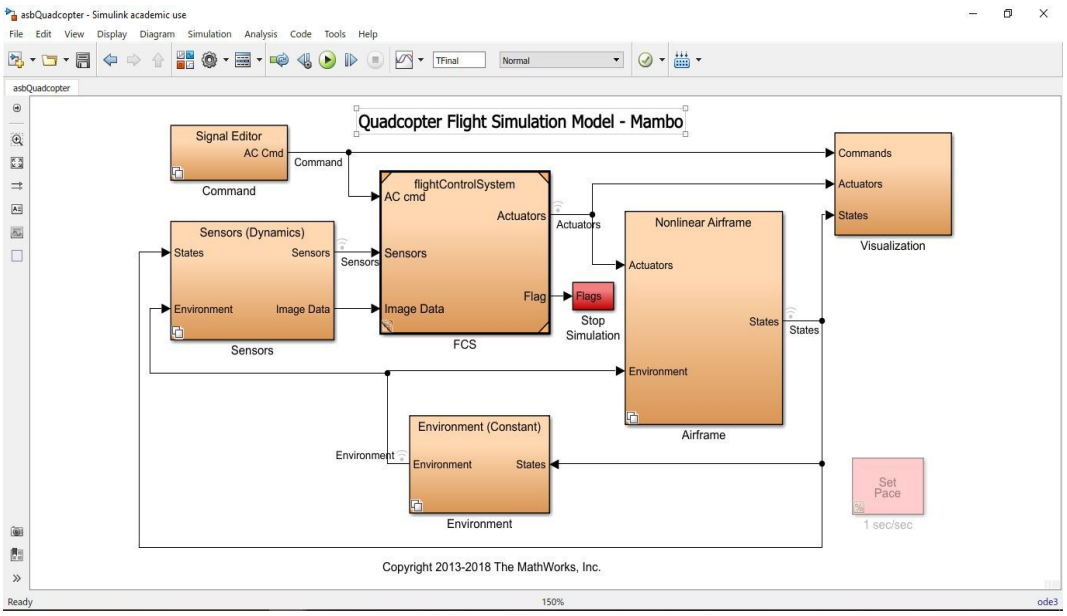


Figura 4.2: Progetto asbQuadcopter su Simulink

distinzione che va fatta per comprendere il funzionamento della struttura è quella tra blocchi che servono per simulare il comportamento del sistema al calcolatore e blocchi che vengono poi processati dall' *Embedded Coder* ed eseguiti sul drone. I blocchi Command, Sensors, Environment, Airframe e Visualization hanno lo scopo, operando assieme, di permettere la visualizzazione al calcolatore del comportamento del drone e sono un utile mezzo per testare gli algoritmi ed il sistema di controllo prima di effettuare il *deploy* del codice sul drone. Il blocco FCS (acronimo che sta per Flight Control System - Sistema di controllo del volo) è l'unico che, attraverso l' *Embedded Coder* "raggiungerà" il drone. Si analizzerà ora, in maniera macroscopica, ciascuna delle componenti/blocchi di cui sopra.

4.2.1 Command

In questo blocco è possibile stabilire le modalità con le quali dare al drone i riferimenti da seguire.

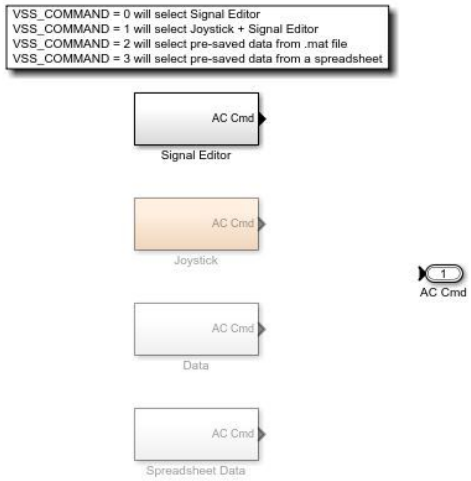


Figura 4.3: Le diverse possibilità per generare riferimenti per il drone

Queste infatti sono molteplici:

- Signal editor: consente di imporre i riferimenti al drone tramite il *Signal Builder* di MATLAB, utile strumento per la creazione di segnali custom;
- Joystick: consente di non imporre aprioristicamente il riferimento da seguire ma di governare attraverso la tastiera del PC (o con l'ausilio di un controller esterno) il drone;
- Data: consente di imporre i riferimenti al drone mediante un foglio di calcolo (analogo ad Excel) ma con formato ed estensione proprietaria di MATLAB;
- Spreadsheet Data: consente di imporre i riferimenti attraverso un foglio di calcolo con estensione .xsls (ovvero Excel).

Tra le possibilità presentate, per l'imposizione dei riferimenti, è stata scelta quella del signal editor (la prima soluzione) in quanto, una volta creato il blocco con i riferimenti desiderati, è sufficiente copiarlo all'interno di FCS per imporli anche sul drone reale (e non solo quello "virtuale") in maniera del tutto diretta. La struttura del Signal Editor è quella presentata nella immagine (sotto - numero immagine). Non verrà dettagliata ulteriormente in quanto superfluo ai fini degli obiettivi di controllo.

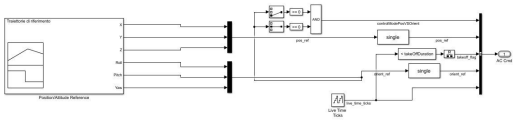


Figura 4.4: Blocco signal editor

4.2.2 Sensor

Questo blocco serve a simulare la presenza dei sensori all'interno del loop (anello) di controllo. Come per il blocco Command sono disponibili più opzioni, in questo caso due.

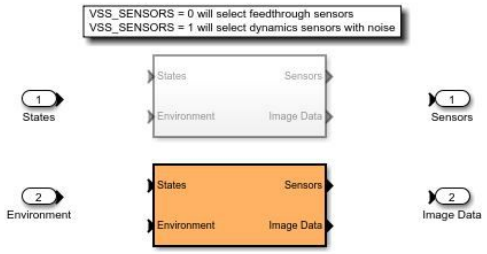


Figura 4.5: Le due possibili scelte nella rappresentazione dei sensori

La prima (feedthrough sensors) consiste nella scelta di sensori ideali che non presentano rumore nè incertezze di misura; la seconda (dynamic sensors) consiste in sensori più vicini a quelli reali che si trovano nel drone, caratterizzati da incertezze di misura e da rumore. Per quanto riguarda questo blocco, la scelta è ricaduta sulla seconda opzione, in modo tale da avere anche in simulazione un comportamento più simile a quello reale.

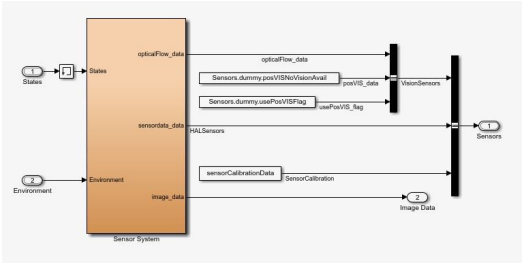


Figura 4.6: Blocco rappresentante i sensori "dinamici"

4.2.3 Environment

Questo blocco serve a simulare l'ambiente all'interno del quale si muove il drone. Come per gli altri blocchi, anche in questo caso sono presenti più opzioni, due in particolare.

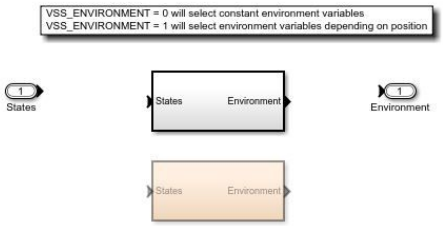


Figura 4.7: Le due possibilità nella scelta della rappresentazione dell'ambiente

La prima opzione (constant environment) modella l'ambiente virtuale come costante, indipendentemente da dove ci si dovesse trovare (non dipendente quindi dalla presenza di ostacoli o edifici). Chiaramente ciò comporta una semplificazione della realtà in quanto la prossimità a edifici o ad ostacoli può mutare l'interazione delle forze aerodinamiche tra il

drone e l'ambiente stesso. La seconda opzione (variable environment) è una rapprsentazione più realistica della realtà, che non fa affidamento sulle semplificazioni di cui sopra. Nel caso in esame, è stato scelta una rappresentazione costante dell'ambiente, in quanto la maggior parte del lavoro è stato svolto sul drone in *hovering* o perlomeno vicino alla posizione di partenza e comunque lontana da edifici e ostacoli di altra natura.

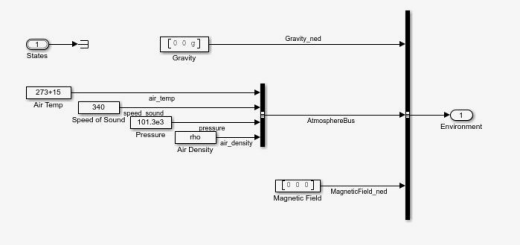


Figura 4.8: Blocco rappresentante l'ambiente costante

4.2.4 Airframe

Questo blocco simula il drone, ovvero il processo di cui si ha interesse governare il comportamento. Qui sono presenti due rappresentazioni del drone, una lineare ed una non lineare.

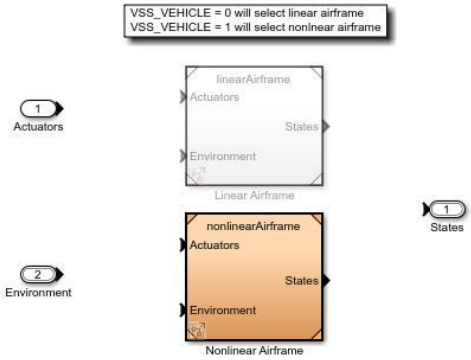


Figura 4.9: Le due possibilità di rappresentazione della struttura del drone

La rappresentazione lineare del processo è adatta per tutte quelle tecniche di controllo che sfruttano per l'appunto la linearità del modello: PID, luogo delle radici, sintesi in frequenza et cetera. Poiché il sistema di controllo che è stato sviluppato è di tipo non lineare, si è scelto, anche per la simulazione, una rappresentazione non lineare del drone e dunque una maggiore fedeltà alla realtà.

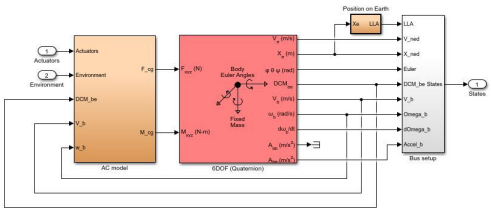


Figura 4.10: Blocco nonlineare rappresentante il drone

4.2.5 Visualization

In questo blocco, che è l'ultimo tra quelli che sono utili esclusivamente alla simulazione del sistema, confluiscono tutti gli elementi di cui sopra (oltre che al FCS che, sebbene sia l'unico ad essere eseguito dal drone, viene comunque eseguito anche in simulazione). Qui si ritrova quanto detto in 4.1: è possibile infatti visualizzare una interfaccia simile a quella presente sul cockpit (plancia) di un aeroplano per la visualizzazione delle singole variabili di interesse (stati del sistema) come:

- Altitudine (in metri e piedi);
- Velocità in nodi;
- Orientamento rispetto al nord in gradi;
- Indicatore di velocità di salita in piedi al minuto;
- Quattro indicatori (uno per motore) di velocità dei motori espressa in RPM (Rounds per minute - giri al minuto).

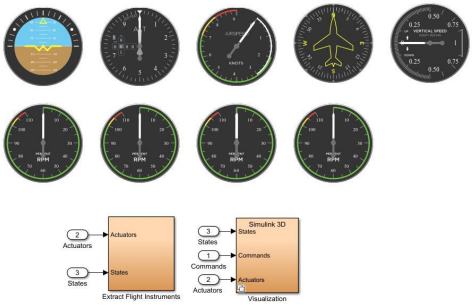


Figura 4.11: Cockpit di Simulink

Scendendo ad un livello più basso all'intero dei blocchi presenti in 4.11 è possibile infine trovare il blocco di *Simulink 3D Animation*, grazie al quale tutte queste informazioni sono raccolte per presentare all'utente una virtualizzata rappresentazione della realtà, come viene mostrato in 4.12

Dopo aver illustrato macroscopicamente il funzionamento del software utilizzato per la simulazione al calcolatore, viene mostrato nel capitolo seguente il nucleo di quest'ultimo, ovvero il controllore che è stato implementato e come questo vi si inserisce.

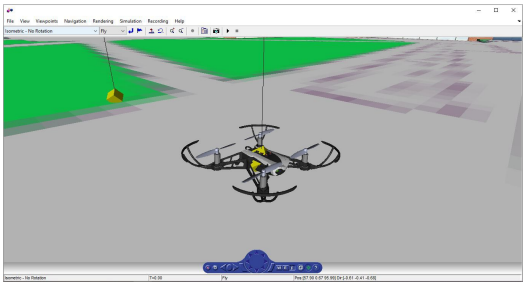


Figura 4.12: Ambiente 3D creato da *Simulink 3D Animation*

4.3 Flight Control System

In questa sezione viene analizzato il blocco (e tutti i suoi "sotto-blocchi") Flight Control System (FCS), il quale è l'unico ad essere convertito in codice *C* ed eseguito sull'hardware del drone. Come si può notare dalla figura 4.13, che isola il blocco della logica di controllo del drone rispetto agli altri componenti, in input vi sono:

- i riferimenti imposti dall'utente (AC cmd);
- le letture provenienti dai sensori (accelerometro, giroscopio, sensore ad ultrasuoni e di pressione);
- le informazioni provenienti dalla telecamera verticale da 60 FPS (Image data).

In output vi sono:

- gli attuatori ovvero i quattro motori del drone;
- un *flag* che, raccogliendo le letture dei sensori ha il compito di interrompere il volo del drone nel caso in cui si verifichino condizioni di pericolo (inclinazione del drone rispetto al terreno oltre una soglia predefinita oppure urti con il terreno o altri oggetti/ostacoli.

Dopo aver visto come il blocco contenente la logica di controllo si interfaccia con le altre componenti del programma si procede alla sua analisi dettagliata.

La figura 4.14 dettaglia ulteriormente la struttura del Flight Control System. É infatti composta dai seguenti blocchi:

- **AC Cmd**: in questo blocco (il cui contenuto è quello di figura 4.4) sono presenti i riferimenti che verranno imposti al drone in fase di volo. Potrebbe però sembrare contraddittorio rispetto a quanto affermato in precedenza in quanto nella figura 4.13 già vi erano in input al blocco FCS dei riferimenti da assegnare il sistema. La presenza di questo blocco in questa posizione permette però che i riferimenti dichiarati siano effettivamente assunti dal drone "reale" e non da quello simulato durante le simulazioni. Si nota inoltre come l'input proveniente dal livello superiore del blocco riguardante i riferimenti sia terminato, ovvero non sia più utilizzato e ciò a conferma di quanto detto fino ad ora. É come se, ancora, il blocco dei riferimenti di figura 4.3 e 4.4 servisse per assegnare i riferimenti per le simulazioni al calcolatore mentre quello di figura 4.14 servisse invece per assegnare i riferimenti al drone "reale". Da queste considerazioni emerge inoltre che sta all'utente il compito di mantenere la coerenza tra questi due blocchi (uno interno e l'altro esterno) per evitare che in simulazione il drone segua delle traiettorie diverse da quelle che magari sono state imposte al drone reale e viceversa;

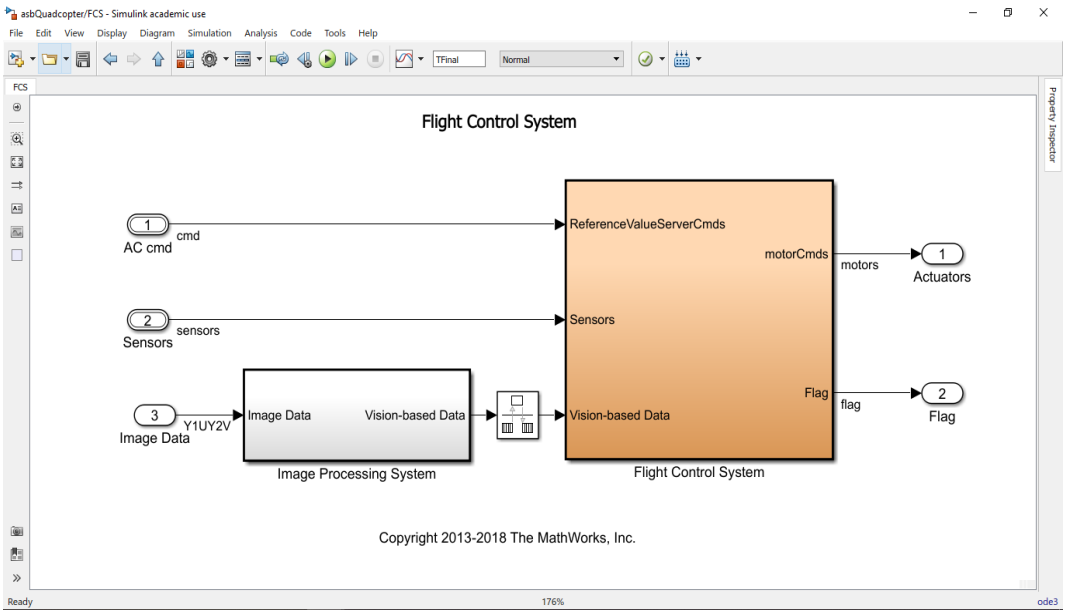


Figura 4.13: FCS - Flight Control System

- **sensordata_group**: raccoglie in esso tutti i dati provenienti dai sensori presenti sul drone. Si può affermare come questi dati siano "grezzi", nel senso che verranno poi rielaborati ed utilizzati nel blocco a valle;

- **estimator**: questo blocco, attraverso l'utilizzo di filtri complementari, filtri di Kalman e *preprocessing* dei dati provenienti dai sensori, realizza la stima delle seguenti variabili, che sono di interesse per l'utilizzatore e fondamentali per la successiva implementazione egli algoritmi di controllo:
  - altitudine e sua derivata nel tempo;
  - posizione del piano xy e sua derivata nel tempo;
  - angoli di roll pitch e yaw e rispettive derivate nel tempo.

- **crash predictor flags**: come emerge dal nome stesso, questo blocco effettua un controllo su alcune particolari condizioni del sistema (posizione del drone e sua velocità) per riuscire a prevenire eventuali voli fallimentari che potrebbero sfociare in comportamenti non desiderati del sistema come ad esempio cadute accidentali o ribaltamenti; **logging**: questo blocco prende in input le informazioni sugli stati del sistema durante la sessione di vole e le salva al suo interno, rendendo possibile una loro analisi offline (ovvero successiva alla sessione di volo stessa). L'analisi dei dati di volo è fondamentale in quanto permette di poter visualizzare graficamente le prestazioni degli algoritmi di controllo ed effettuare una loro valutazione secondo criteri oggettivi.

- **controller**: all'interno di questo blocco è presente il vero e proprio algoritmo di controllo che è stato implementato nel del presente lavoro di tesi. Ai fini di maggior chiarezza espositiva, verrà dedicata a questo sottosistema la sottosezione successiva.

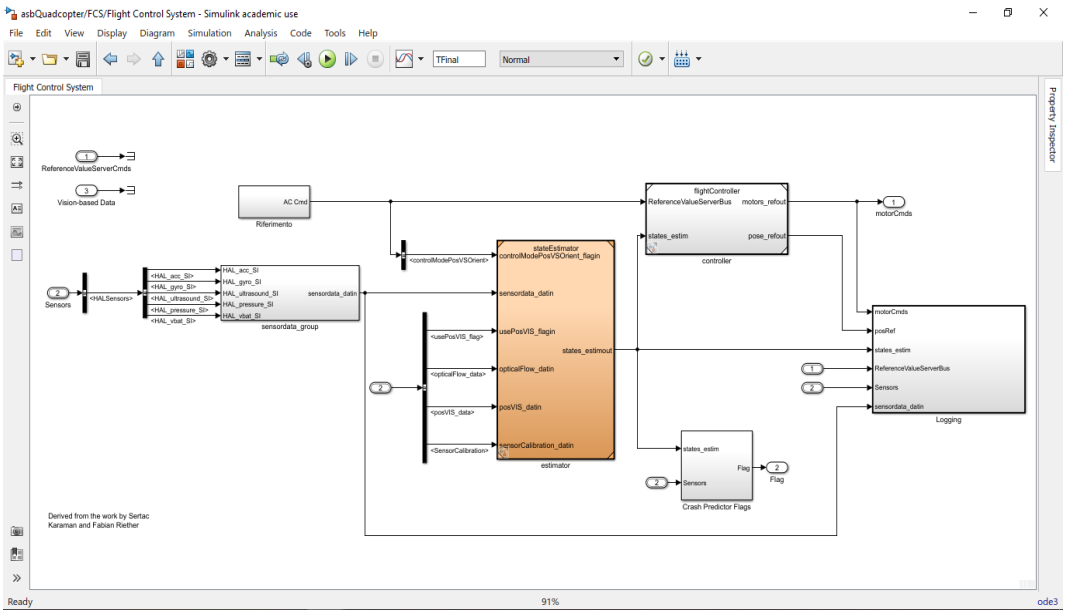


Figura 4.14: Struttura del Flight Control System

4.3.1 Flight Controller

Andando ad isolare, rispetto allo schema presente in figura 4.14, il controllore di volo si può notare, come mostrato in figura 4.15, che gli ingressi del controllore sono i riferimenti e la stima degli stati del sistema e le uscite sono i segnali di controllo dei quattro motori del drone ed i riferimenti in posizione ed orientamento, i quali andranno a confluire nel blocco del logging dei dati nel livello superiore dello schema.

Scendendo ulteriormente di un livello emerge finalmente la struttura dello schema di controllo. Macroscopicamente si può affermare che a partire dai riferimenti e dal confronto di questi ultimi con i segnali di riferimento (valutando quindi l'errore) viene calcolato per l'altitudine e per gli angoli di roll, pitch e yaw lo sforzo di controllo opportuno per portare a zero questo errore. Lo sforzo di controllo calcolato finisce poi nel blocco di colore verde chiamato motor mixing algorithm. All'interno di questo blocco sono riportate le considerazioni svolte nella sezione 3.1.3: a partire dallo sforzo di controllo (nello specifico alla spinta che il motore deve fornire) ed al movimento nello spazio tridimensionale che il drone deve compiere vengono ripartite le spinte che il singolo motore deve compiere. A valle del blocco del motor mixing algorithm vi è un blocco il cui compito è quello di convertire lo sforzo di controllo espresso in spinta a segnale di controllo adatto al singolo motore. Questa trasformazione di grandezze avviene attraverso la moltiplicazione per specifiche costanti che sono caratteristiche del singolo motore e che dipendono dalla sua struttura e modalità costruttiva. A monte dei blocchi in verde vi sono i controllori. Disaccoppiando in qualche modo il movimento nello spazio in raggiungimento di una determinata quota di volo e spostamento successivo nello spazio bidimensionale si possono analizzare i controllori e la loro organizzazione. Il controllore di yaw e quello di altitudine sono quelli concettualmente più semplici: a partire dall'errore tra il riferimento ed il presente stato del sistema calcolano lo sforzo di controllo adatto per far sì che il sistema insegua correttamente il riferimento desiderato. Per quanto riguarda i controllori di pitch e roll, questi presentano una struttura a due livelli. Vi è un controllore esterno di tipo PID (Proporzionale, Integrale, Derivativo) che, considerato

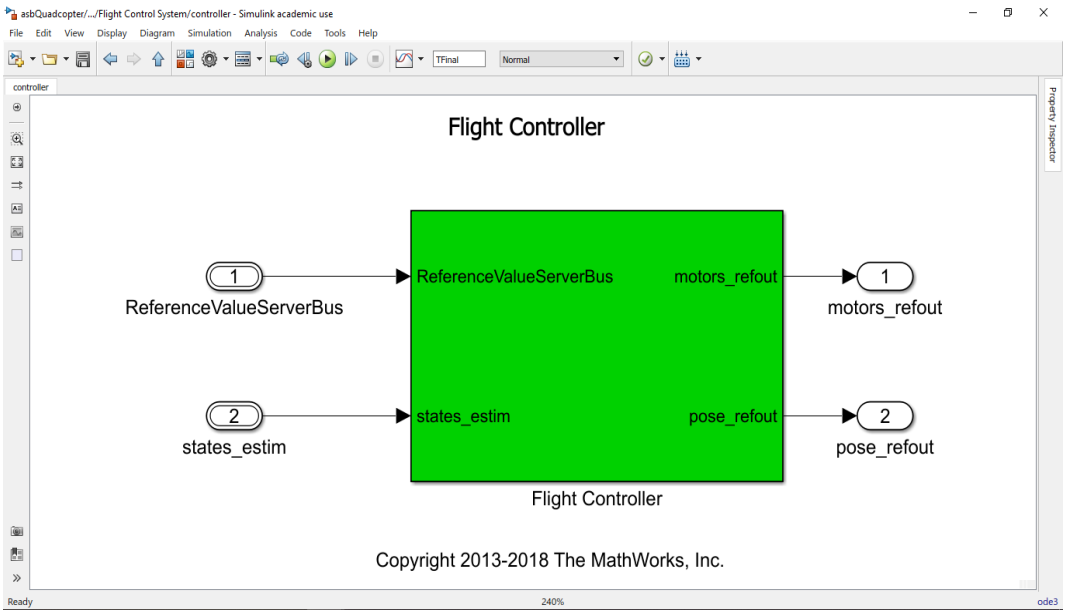


Figura 4.15: Flight Controller

l'errore di posizione del drone (errore calcolato sul piano xy) fornisce gli opportuni riferimenti che i controllori interni di pitch e roll devono inseguire. Ciò è necessario se, ad esempio, si considera la seguente situazione. Si immagini che il drone sia in volo stazionario rispetto al terreno ad una quota fissa e che non via sia il livello esterno di controllo di posizione. Da quanto appena detto i riferimenti per il pitch e per il roll devono essere obbligatoriamente nulli. Si immagini ora che un disturbo (ad esempio una folata di vento laterale) perturbi il sistema dal suo stato stazionario. I controllori correggeranno la posizione angolare del drone riportandolo in una posizione stazionaria rispetto al terreno senza però avere memoria della posizione di partenza del drone. Se inoltre si immagina che il disturbo sul sistema si ripeta nel corso del tempo si può immediatamente dedurre come nel lungo periodo ciò potrebbe portare il volo del drone ben lontano dalla posizione di partenza ed in generale dalla traiettoria in posizione che il drone si desidera che segua. Giustificata quindi la presenza del loop esterno di controllo in posizione va aggiunta una nota riguardo il calcolo della velocità dei motori. Negli algoritmi di controllo dei capitoli 5 e 6, uno degli elementi necessari al calcolo dello sforzo di controllo è la velocità dei motori espressa in radianti al secondo. Il *Parrot Mambo* non è però fornito di sensori (encoder) per il calcolo di questa grandezza fisica. Per risolvere tale problema implementativo si è sfruttata la struttura Simulink preesistente. Nel blocco verde a valle del motor mixing algorithm è infatti presente il valore, calcolato in base agli algoritmi di controllo, del numero di giri al secondo che ciascun motore deve compiere. Ritardando di un passo di campionamento tale informazione e moltiplicandola per 2π è stato dunque possibile far fronte a tale problema, introducendo nello schema di controllo il blocco chiamato "calcolo termine ΩJ_{tp}" che svolge le operazione sopra menzionate. Quanto fino a ciò espresso è visibile nella figura 4.16. Scendendo infine di un ulteriore livello è possibile analizzare la struttura del singolo controllore. A titolo esemplificativo è mostrato, nella figura 4.17 il controllore per l'angolo di yaw: all'interno del blocco sono implementate le equazioni 5.18 e 6.7 relative all'angolo di yaw ed esternamente al blocco è presente la struttura per il calcolo dell'indice prestazionale IAE di cui al capitolo 7. Struttura del tutto analoga è valida per gli altri controllori di altitudine e degli angoli di pitch e yaw.

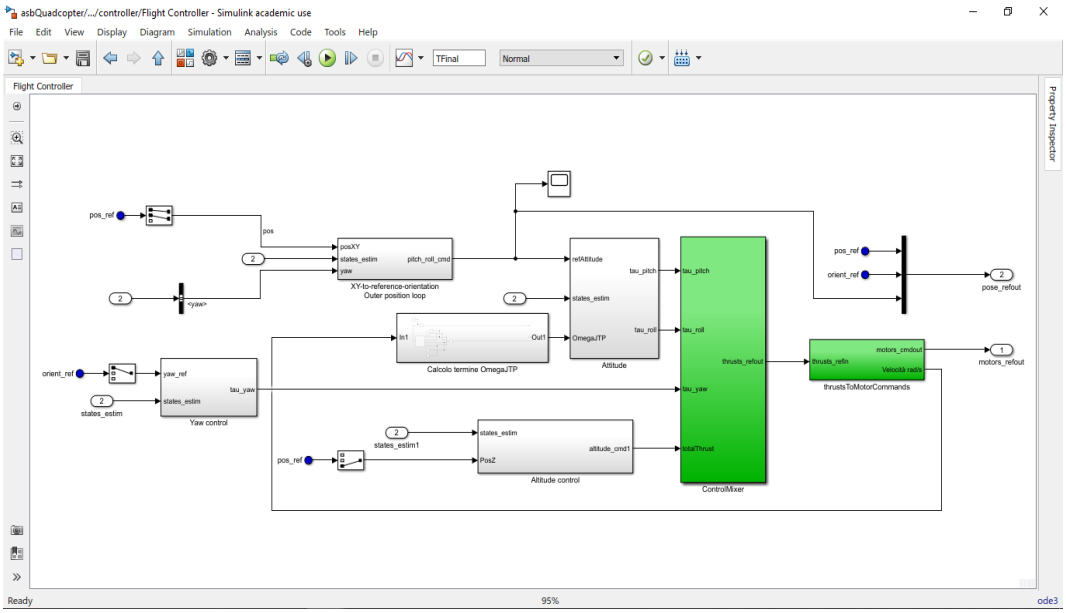


Figura 4.16: Struttura del flight controller

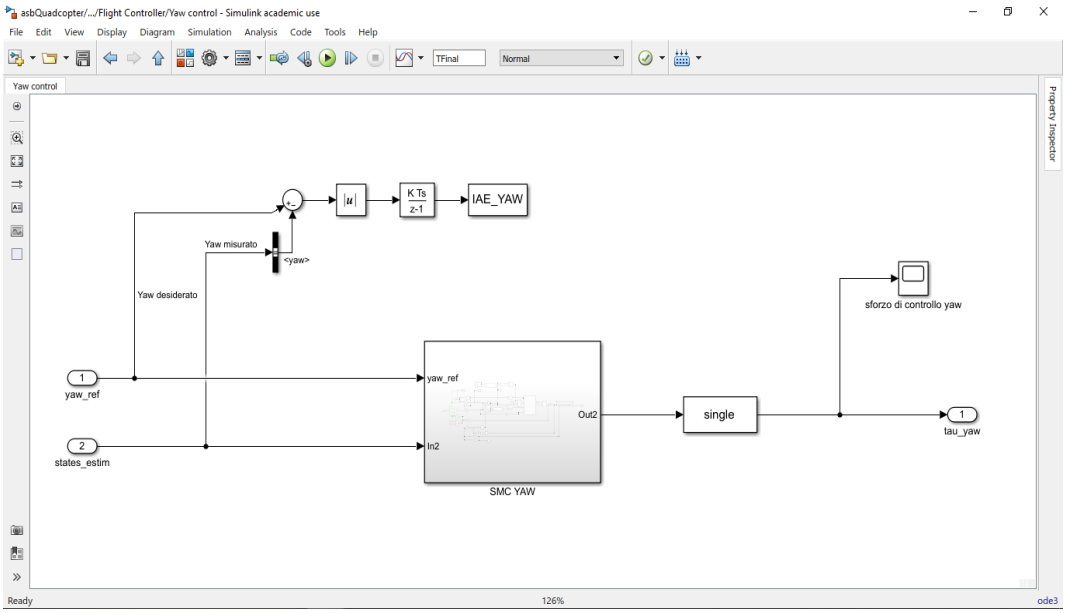


Figura 4.17: Controllore per l'angolo di yaw

Capitolo 5

Controllore sliding mode

5.1 Considerazioni generali

La parola *controllo*, nell'uso tecnico che qui interessa, ha un significato diverso da quello corrente nella lingua italiana (nella quale assume più una accezione di *verifica*) ed è più affine a quello della parola *control* della lingua inglese. Per controllo, infatti, si intende l'azione o l'insieme delle azioni indirizzate a far assumere ad una grandezza, in generale una grandezza fisica, un valore determinato o una successione determinata di valori nel tempo. Così, ad esempio, per "controllo della velocità di un motore" si intende il complesso delle azioni intese a far assumere a questa velocità un valore prefissato oppure a far sì che l'evoluzione nel tempo del valore di questa velocità corrisponda ad una prefissata funzione. Si consideri ora un determinato oggetto o processo, naturale o artificiale e lo si denomini come *oggetto o processo controllato*. Si assuma inoltre che in esso siano state individuate alcune grandezze delle quali è di interesse l'evoluzione nel tempo. Se vi sono dunque grandezze delle quali è di interesse che abbiano un andamento prefissato nel tempo, si è indotti ad introdurre ed a mettere in evidenza nell'oggetto o nel processo altre grandezze, manipolando le quali sia possibile ottenere lo scopo desiderato. In questo modo si viene a considerare l'oggetto o processo in questione da un punto di vista sostanzialmente analogo a quello in cui ci si pone per associare un sistema astratto orientato. L'orientamento in un sistema astratto corrisponde alla suddivisione delle variabili in causa ed effetti e naturalmente suggerisce l'immagine di un sistema dinamico - che evolve nel tempo - come una scatola nera, che rappresenta le modalità secondo le quali le variabili di uscita sono influenzate da quelle di ingresso. [12]

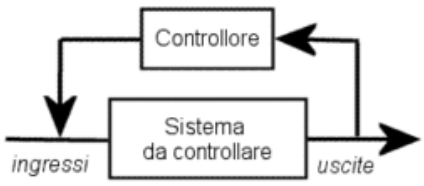


Figura 5.1: Schema di controllo dove è possibile osservare: ingressi, uscite, sistema e controllore

- Le grandezze di cui si interessa osservare l'evoluzione si presentano come *uscite* e vengono dette grandezze controllate;

- le grandezze che possono venire manipolate allo scopo di ottenere le evoluzioni volute si presentano come *ingressi* e vengono dette grandezze controllanti o di controllo

- l'oggetto o processo in grado di imporre alle grandezze controllanti gli andamenti prestabiliti prende il nome di oggetto o processo controllante o, più semplicemente, di *controllore*

5.2 Controllo sliding mode

Il *controllo sliding mode* consiste in una legge di controllo che commuta molto velocemente e che forza le traiettorie di stato a raggiungere una superficie, detta per l'appunto *superficie di sliding*, in tempo finito e a farle rimanere su tale superficie per gli istanti di tempo futuri: le traiettorie di stato raggiungono quindi la *superficie di sliding* in un tempo T_0 e poi vi rimangono, "scivolando" (sliding) sulla medesima per gli istanti di tempo $t \geq T_0$ [13].

Il *design* (progettazione) di una legge di controllo di tipo sliding mode è una proceduta a due passi:

- sintesi (scelta) della superficie di sliding, la quale imporrà le performace desiderate;

- design della legge di controllo, la quale assicura che tutte le traiettorie di stato raggiungano e poi "scivolino" sulla superficie di sliding.

L'algoritmo di controllo sliding mode si compone di sue parti: il *controllo equivalente* ed il *controllo correttivo* [14]. Il controllo equivalente serve ad assicurarsi che le traiettorie raggiungano la superficie e che vi rimangano, mantenendo la derivata nel tempo della superficie pari a 0. Il controllo correttivo dall'altra parte serve a compensare qualsiasi variazione attorno alla superficie, sia se dovuta a disturbi esterni sia se dovuta a delle dinamiche non modellate del sistema:

$$U_i = U_i^{eq} + U_i^C \tag{5.1}$$

dove U_i è la legge di controllo per un dato stato del sistema, U_i^{eq} è il controllo equivalente e U_i^C è il controllo correttivo.

Sia ora la superficie di sliding del tipo [15]:

$$S_i = \dot{e}_i + \alpha_i e_i \tag{5.2}$$

dove S_i è la superficie di sliding per un dato stato i del sistema, e_i sia l'errore tra lo stato misurato x ed il suo valore di riferimento/desiderato x^d e α_i un termine che rappresenta un guadagno di tipo proporzionale.

Si consideri ora la seguente funzione di Lyapunov:

$$V_i = \frac{1}{2} S_i^2 \tag{5.3}$$

Per assicurarsi che il sistema sia stabile, la derivata rispetto al tempo della funzione di Lyapunov deve essere strettamente negativa. Si deve quindi avere che l'equazione 5.4:

$$\dot{V}_i = S_i \dot{S}_i \tag{5.4}$$

abbia derivata rispetto al tempo strettamente negativa. Per fare in modo che l'equazione 5.4 rispetti tale condizione si può porre [16]:

$$\dot{S}_i = -\beta_i S_i - \lambda_i \operatorname{sign}(S_i) \tag{5.5}$$

Moltiplicando entrambi i termini (destro e sinistro) per S_i si ha:

$$S_i \dot{S}_i = -\beta_i S_i^2 - \lambda_i |S_i| \tag{5.6}$$

che è strettamente negativa per $\beta_i > 0$ e $\lambda_i > 0$.

5.2.1 Controllo equivalente

Imponendo che le derivate rispetto al tempo di ciascuna superficie sia uguale a 0, facendo riferimento al sistema di equazioni 3.16 ed isolando i termini del tipo U_i si ottiene che:

- *Altitudine*:

$$\dot{S}_z = \ddot{e}_z + \alpha_z \dot{e}_z = (\ddot{Z}^d - \ddot{Z}) + \alpha_z \dot{e}_z = \ddot{Z}^d + g - \cos\theta \cos\phi \frac{U_z}{m} + \alpha_z \dot{e}_z \tag{5.7}$$

  Imponendo in 5.7 $\dot{S}_z = 0$ si ottiene:

$$U_z^{eq} = \frac{m}{\cos\theta \cos\phi}(\ddot{Z}^d + g + \alpha_z \dot{e}_z) \tag{5.8}$$

Procedendo in modo analogo per le superfici di sliding per il *Roll*, il *Pitch* e lo *Yaw* si ottengono i seguenti controlli equivalenti:

- *Roll*:

$$U_\phi^{eq} = J_{xx}(\ddot{\phi}^d + \alpha_\phi \dot{e}_\phi) - (J_{yy} - J_{zz})\dot{\theta}\dot{\psi} - \dot{\theta}\Omega J_{tp} \tag{5.9}$$

- *Pitch*:

$$U_\theta^{eq} = J_{yy}(\ddot{\theta}^d + \alpha_\theta \dot{e}_\theta) - (J_{zz} - J_{xx})\dot{\phi}\dot{\psi} - \dot{\phi}\Omega J_{tp} \tag{5.10}$$

- *Yaw*:

$$U_\psi^{eq} = J_{zz}(\ddot{\psi}^d + \alpha_\psi \dot{e}_\psi) - (J_{xx} - J_{yy})\dot{\phi}\dot{\theta} \tag{5.11}$$

5.2.2 Controllo correttivo

Per il calcolo del controllo correttivo si parte dal riscrivere la rappresentazione in spazio di stato di 3.16 come:

$$\dot{x} = f(x) + g(x)u \tag{5.12}$$

Per soddisfare la condizione di stabilità di Lyapunov espressa dall'equazione 5.5, si prende il generico controllo correttivo U_i^C del tipo:

$$U_i^C = g_i(x)^{-1}(\beta_i S_i + \lambda_i \operatorname{sign}(S_i)) - f(x) \tag{5.13}$$

Applicando l'espressione 5.13 agli angoli di roll, pitch, yaw e al controllo di altitudine, si trova il seguente controllo correttivo:

- *Altitudine*:
$$U_z^C = \frac{m}{\cos\phi\cos\theta}(\beta_z S_z + \lambda_z \operatorname{sign}(S_z)) \tag{5.14}$$
- *Roll*:
$$U_\phi^C = J_{xx}(\beta_\phi S_\phi + \lambda_\phi \operatorname{sign}(S_\phi)) \tag{5.15}$$
- *Pitch*:
$$U_\theta^C = J_{yy}(\beta_\theta S_\theta + \lambda_\theta \operatorname{sign}(S_\theta)) \tag{5.16}$$
- *Yaw*:
$$U_\psi^C = J_{zz}(\beta_\psi S_\psi + \lambda_\psi \operatorname{sign}(S_\psi)) \tag{5.17}$$

5.2.3 Controllo complessivo

Per quanto detto fino ad ora e dall'espressione in 5.1 segue che la legge di controllo complessiva, ovvero formata dal controllo equivalente e dal controllo correttivo, per l'altitudine e gli angoli di roll, pitch e yaw è la seguente:

$$\begin{cases} U_z = \frac{m}{\cos\theta\cos\phi}(\ddot{Z}^d + g + \alpha_z\dot{e}_z + \beta_z S_z + \lambda_z \operatorname{sign}(S_z)) \\ U_\phi = J_{xx}(\ddot{\phi}^d + \beta_\phi S_\phi + \lambda_\phi \operatorname{sign}(S_\phi) + \alpha_\phi \dot{e_\phi}) - (J_{yy} - J_{zz})\dot{\theta}\dot{\psi} - \dot{\theta}\Omega J_{tp} \\ U_\theta = J_{yy}(\ddot{\theta}^d + \beta_\theta S_\theta + \lambda_\theta \operatorname{sign}(S_\theta) + \alpha_\theta \dot{e_\theta}) - (J_{zz} - J_{xx})\dot{\phi}\dot{\psi} + \dot{\phi}\Omega J_{tp} \\ U_\psi = J_{zz}(\ddot{\psi}^d + \beta_\psi S_\psi + \lambda_\psi \operatorname{sign}(S_\psi) + \alpha_\psi \dot{e_\psi}) - (J_{xx} - J_{yy})\dot{\phi}\dot{\theta} \end{cases} \tag{5.18}$$

Capitolo 6

Controllo robusto

6.1 Disturbi

Un disturbo è, in generale, costituito da un segnale indesiderato che agisce sul sistema modificandone il comportamento dinamico e quindi l'andamento dell' uscita. I disturbi possono essere generati da svariate cause, fra cui ad esempio [17]:

- Il rumore dei dispositivi elettronici interni al sistema da controllare e/o presenti negli attuatori e trasduttori impiegati
- Elementi "fisici" del sistema, quali il carico variabile di un braccio robotico, il vento per un'antenna radar, l'attrito in un sistema meccanico, ecc.
- Si possono considerare come disturbi, con una accezione più generica, anche dinamiche non modellate del sistema oppure forze che agiscono sul sistema che sono state trascurate.

Un controllo che è quindi definito "robusto" (si veda 6.2) è quello che riesce strutturalmente a sopperire ed a compensare ,in qualche modo, questi disturbi.

6.2 Robustezza del controllo

Riprendendo quanto detto nella sezione precedente, il termine robustezza indica la capacità di un sistema di controllo di garantire stabilità e precisione anche in presenza di variazioni incognite di parametri che caratterizzano il modello del sistema. I tipi di variazioni che usualmente si considerano nell' ambito di questa disciplina possono essere distinti in due famiglie che prendono il nome di:

- Incertezze strutturate
- Incertezze non strutturate

Le prime sono quelle dovute a parametri che occupano un posto ben definito, che svolgono un ruolo ben preciso all'interno del modello. Si pensi ad esempio ad una costante di tempo di un circuito elettrico, ad un guadagno di un amplificatore, ad un coefficiente d'attrito in un motore. Questi parametri sono noti in un intervallo di tolleranza ben preciso.
Le seconde rappresentano in qualche modo il caso complementare alle prime. Nel senso che le incertezze non strutturate sono perturbazioni dovute a fenomeni non modellati, trascurati. Si pensi ad esempio ad un ritardo temporale oppure ad una elasticità. Questi sono fenomeni dinamici che se considerati nel modello danno luogo ad equazioni pi'u complicate con conseguente aumento dell'ordine del sistema. Oppure non si è in grado di rappresentare

matematicamente tali fenomeni. Trascurarli può dar luogo ad inconvenienti in sede di risultati e a prestazioni non soddisfacenti. Le incertezze non strutturate tengono quindi in conto gli effetti di parti di modello che non vengono considerate. [18]

6.3 Controllore robusto

Il controllo sliding mode è costruito in modo tale da essere naturalmente robusto per disturbi limitati. Il termine contente la funzione segno (si veda il sistema di equazioni 5.18) dà all'algoritmo di controllo una forte capacità di gestire incertezze sul modello, dinamiche non modellate e disturbi esterni. Lo stesso termine però è responsabile del fenomeno del *chattering* ed introduce discontinuità nell'azione di controllo. Si prenda ora l'equazione 5.5:

$$\dot{S}_i = -\beta_i S_i - \lambda_i \operatorname{sign}(S_i)$$

Si introduce ora il termine d_i, che rappresenta tutti i disturbi dovute ad incertezze sul modello ed a dinamiche non modellate. La dinamica della superficie di sliding (la sua derivata nel tempo) assume la seguente forma:

$$\dot{S}_i = U_i + d_i \tag{6.1}$$

dove $U_i = -\beta_i S_i - \lambda_i \operatorname{sign}(S_i)$ rappresenta la legge di controllo (come ad esempio quelle presenti in 5.18). Per garantire che il sistema sia stabile, il termine λ_i deve essere in grado di compensare il disturbo d_i. Si deve quindi avere che:

$$\lambda_i > \|d_i\|_\infty$$

Un'assunzione di questo tipo sulla limitatezza uniforme (*uniform boundedness*) di dinamiche non modellate e di disturbi esterni è però piuttosto restrittiva e non sempre soddisfatta, per questo motivo già in 5.5 è presente un termine lineare ($\beta_i S_i$) che serve a compensare disturbi con un limite superiore sconosciuto o che crescono proporzionalmente con S_i. In ogni caso, per un quadricottero, una più realistica rappresentazione della struttura dei disturbi di cui sopra, li vede scritti come segue [16]:

$$d_i = \rho_i |S_i|^{p_i} |U_i|^{v_i} + d_{0i} \tag{6.2}$$

dove $0 < v_i < 1$, $0 \leq \rho_i \leq 1$, p_i è un numero reale positivo e d_{0i} è un disturbo limitato in modulo. Da queste considerazione segue una modifica della dinamica della superficie di sliding e di conseguenza una modifica non lineare dell'algoritmo di controllo. Da $U_i = -\beta_i S_i - \lambda_i \operatorname{sign}(S_i)$ si passa a:

$$U_i = (-\beta_i |S_i|^{m_i} - \lambda_i) \operatorname{sign}(S_i) \tag{6.3}$$

dove m_i è un intero positivo.
La derivata della funzione di Lyapunov diventa quindi:

$$\dot{V}_i = S_i \dot{S}_i = -\beta_i |S_i|^{m_i+1} - \lambda_i |S_i| + \rho_i |S_i|^{p_i} S_i \big(\big| -\beta i |S_i|^{m_i} \operatorname{sign}(S_i) - \lambda_i \operatorname{sign}(S_i) \big| \big)^{v_i} + S_i d_{0i} \tag{6.4}$$

Sapendo che $|a+b|^{v_i} < |a|^{v_i} + |b|^{v_i}$ se $0 < v_i < 1$ si ottiene che:

$$\begin{aligned}\dot{V}_i &\leq -\beta_i|S_i|^{m_i+1} - \lambda_i|S_i| + \rho_i|S_i|^{p_i}\, S_i|\beta_i|\, S_i^{m_i}\big|\text{sign}(S_i)\big|^{v_i} + \rho_i|S_i|^{p_i}\, S_i\big|{-\lambda_i}\,\text{sign}(S_i)\big|^{v_i} + d_{0i}S_i \\ &\leq -\beta_i|S_i|^{m_i+1} - \lambda_i|S_i| + \rho_i\beta_i^{v_i}|S_i|^{m_i v_i + p_i + 1} + \rho_i\lambda_i^{v_i}|S_i|^{p_i+1} + d_{0i}S_i\end{aligned} \tag{6.5}$$

Si ottiene infine che:

$$\dot{V}_i \leq -\beta_i|S_i|^{m_i+1} - \lambda_i|S_i| + \rho_i\beta_i^{v_i}|S_i|^{m_i v_i + p_i + 1} + \rho_i\lambda_i^{v_i}|S_i|^{p_i+1} + d_{0i}S_i \tag{6.6}$$

Poiché $\dot{V}_i$ deve essere strettamente negativa si deve avere che:

$$m_i + 1 > m_i v_i + p_i + 1$$

Scegliendo [16]:

$$m_i = \frac{p_i + 1 - v_i}{1 - v_i}$$

viene soddisfatta la condizione di cui sopra, con v_i e p_i che modellano il disturbo d_i. Segue quindi che l'algoritmo di controllo complessivo è il seguente [16]:

$$\begin{cases} U_z = \frac{m}{\cos\theta\cos\phi}(\ddot{Z}^d + g + \alpha_z\dot{e}_z + \beta_z|S_z|^{\frac{p_z+1-v_z}{1-v_z}}\,\text{sign}(S_z) + \lambda_z\,\text{sign}(S_z)) \\ U_\phi = J_{xx}(\ddot{\phi}^d + \beta_\phi\big|S_\phi\big|^{\frac{p_\phi+1-v_\phi}{1-v_\phi}}\,\text{sign}(S_\phi) + \lambda_\phi\,\text{sign}(S_\phi) + \alpha_\phi\dot{e_\phi}) - (J_{yy} - J_{zz})\dot{\theta}\dot{\psi} - \dot{\theta}\Omega J_{tp} \\ U_\theta = J_{yy}(\ddot{\theta}^d + \beta_\theta|S_\theta|^{\frac{p_\theta+1-v_\theta}{1-v_\theta}}\,\text{sign}(S_\theta) + \lambda_\theta\,\text{sign}(S_\theta) + \alpha_\theta\dot{e_\theta}) - (J_{zz} - J_{xx})\dot{\phi}\dot{\psi} + \dot{\phi}\Omega J_{tp} \\ U_\psi = J_{zz}(\ddot{\psi}^d + \beta_\psi\big|S_\psi\big|^{\frac{p_\psi+1-v_\psi}{1-v_\psi}}\,\text{sign}(S_\psi) + \lambda_\psi\,\text{sign}(S_\psi) + \alpha_\psi\dot{e_\psi}) - (J_{xx} - J_{yy})\dot{\phi}\dot{\theta} \end{cases} \tag{6.7}$$

Capitolo 7

Risultati sperimentali

7.1 Struttura delle simulazioni

Verranno ora presentati i risultati ottenuti dallo sviluppo e dalla implementazione degli algoritmi in 5 e in 6. La struttura del presente capitolo è la seguente:

- Risultati delle simulazioni al calcolatore della durata di 100 secondi: queste simulazioni utilizzano il controllore sliding mode del capitolo 5. Nessun disturbo aggiuntivo è presente nel sistema;
- Risultati delle simulazioni al calcolatore della durata di 10 secondi con il controllore sliding mode del capitolo 5 alle quali viene aggiunto il disturbo d_i del capitolo 6;
- Risultati delle simulazioni al calcolatore della durata di 10 secondi con il disturbo d_i del capitolo 6 ma, al posto del controllore sliding mode del primo punto, il controllore sliding mode robusto del capitolo 6;
- Risultati di prove sperimentali sul sistema reale

Il segnale di riferimento sarà di colore arancione mentre le uscite del drone saranno di colore azzurro.

7.2 Controllore sliding mode non robusto

In questa sezione vengono mostrati i risultati della prima simulazione effettuata. Il controllore utilizzato è quello del capitolo 5 ed in questo caso non è presente il disturbo d_i del capitolo 6. I riferimenti imposti sono i seguenti:

Ovvero, come mostrato nella figura 7.1, si impone che:

- l'altitudine rimanga fissa per tutto il tempo della simulazione ad un valore di $0,8$ metri (il valore in figura è negativo semplicemente perchè l'asse z si considera rivolto verso il basso piuttosto che verso l'alto);
- lo yaw segua l'andamento della curva in viola nell'immagine, ovvero che si compia dapprima un giro su se stessi fino al secondo 20 circa e che poi si compia una rotazione su se stessi nell'altro senso, fino a ritornare con l'imbardata nella posizione di partenza;
- l'andamento lungo l'asse x sia quello mostrato in figura dalla curva verde.

Questi riferimenti sono stati scelti con il fine di far compiere al controllore un moto "vario" e per dimostrare l'efficacia del controllore proposto. Per quanto riguarda l'andamento sull'asse y, il riferimento si è posto pari a zero per tutto il tempo della simulazione. Discorso a parte,

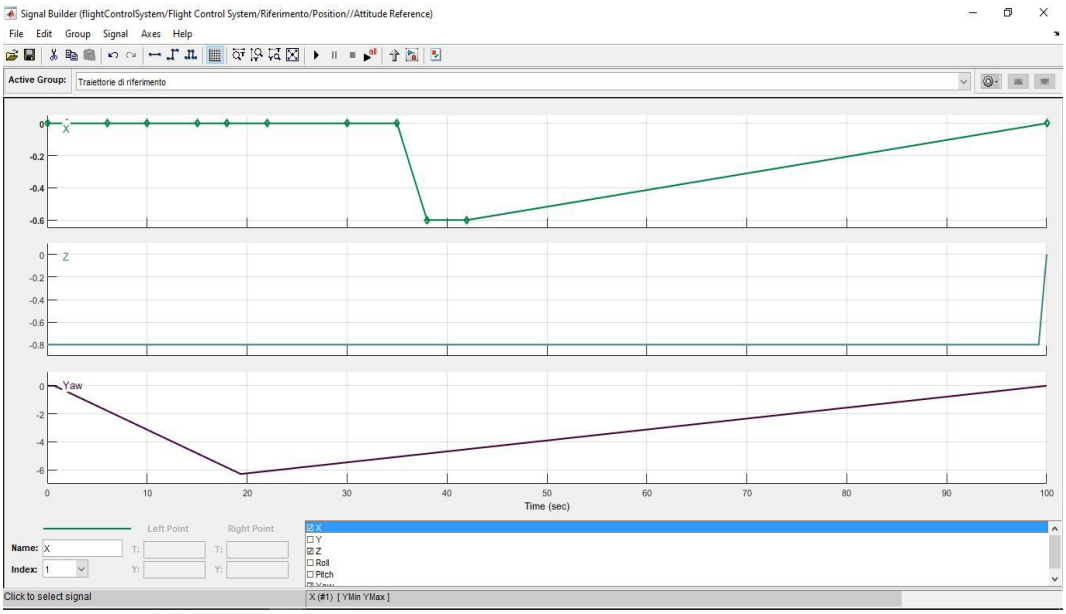


Figura 7.1: Riferimenti per la prima simulazione

come già esposto nel capitolo 4, è per gli angoli di roll e di pitch. Questi non possono essere imposti direttamente, o meglio, il loro riferimento non può essere imposto direttamente dal progettista ma è necessario passare per il loop esterno di controllo di posizione: sarà proprio questo, a seconda del riferimento imposto in posizione, a fornire ai controllori di pitch e roll il riferimento opportuno da seguire. I riferimenti di roll e di pitch non verranno quindi inseriti qui ma verranno mostrati direttamente nelle immagini delle simulazioni. Di seguito, prima di mostrare le immagini dei risultati, viene inserita una tabella che mostra i parametri di progetto ed i valori che vi sono stati assegnati.

Tabella 7.1: Parametri di progetto e loro valore per l'asse z

Parametro	Valore
α_z	20
β_z	1
λ_z	30
ϵ_z	0,1

Tabella 7.2: Parametri di progetto e loro valore per l'angolo di yaw

Parametro	Valore
α_{yaw}	1
β_{yaw}	1
λ_{yaw}	1
ϵ_{yaw}	0,2

Sebbene non si è interessati in questo lavoro al controllo di posizione, si riportano in ogni caso i valori del controllore PID che controlla la posizione del drone:

Tabella 7.3: Parametri di progetto e loro valore per l'angolo di pitch

Parametro	Valore
α_{pitch}	5
β_{pitch}	1
λ_{pitch}	15
ϵ_{pitch}	0,3

Tabella 7.4: Parametri di progetto e loro valore per l'angolo di roll

Parametro	Valore
α_{roll}	5
β_{roll}	1
λ_{roll}	15
ϵ_{roll}	0,3

Tabella 7.5: Parametri di progetto e loro valore per il controllore PID

Parametro	Valore
P_x	$-0,07$
P_y	$+0,07$
I_x	0
I_y	0
D_x	$0,01$
D_y	$-0,01$

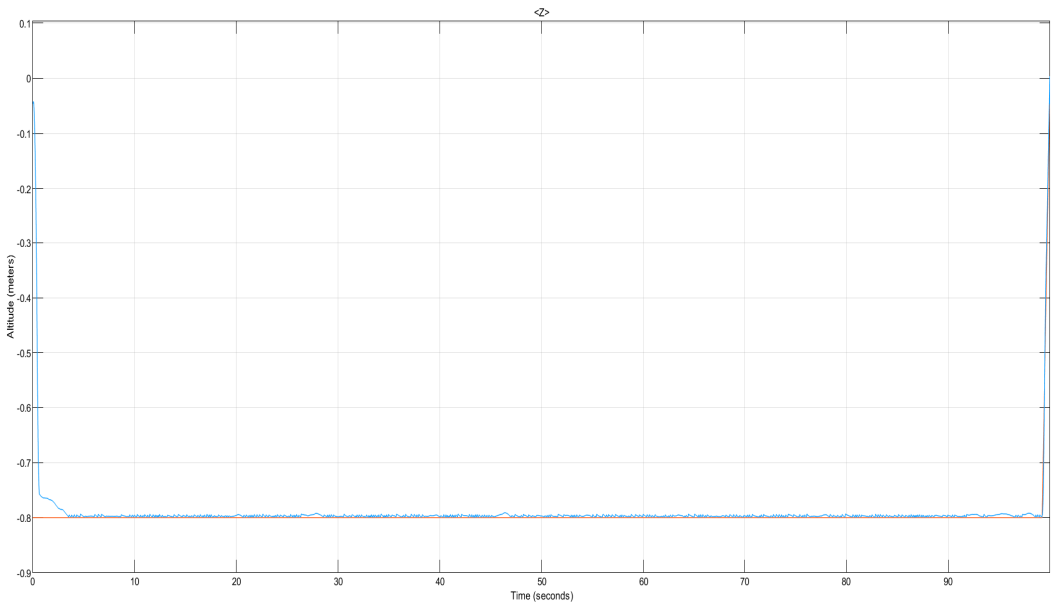


Figura 7.2: Risultato prima simulazione asse z

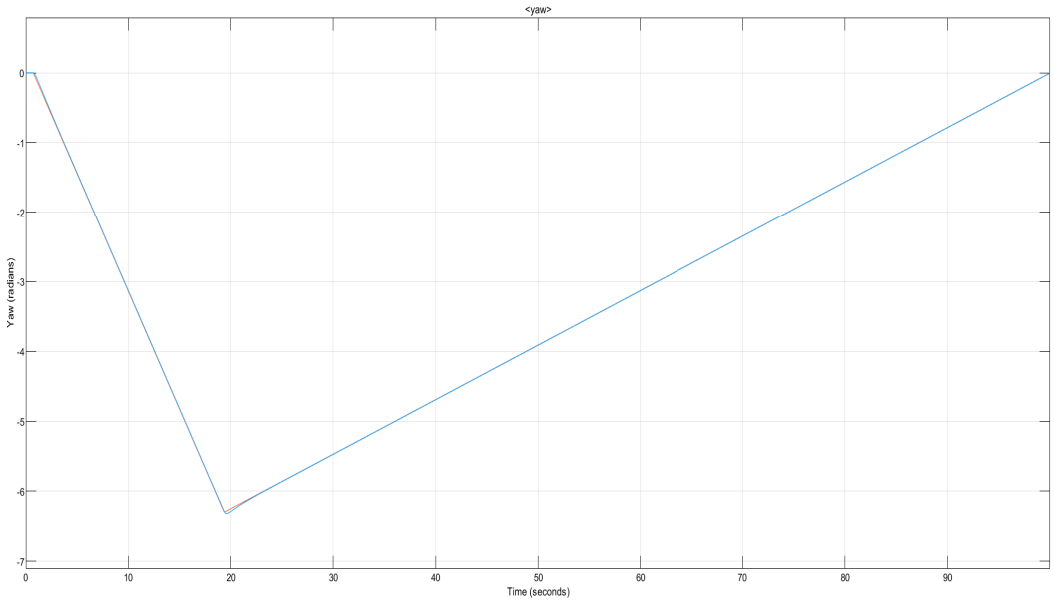


Figura 7.3: Risultato prima simulazione angolo di *yaw*

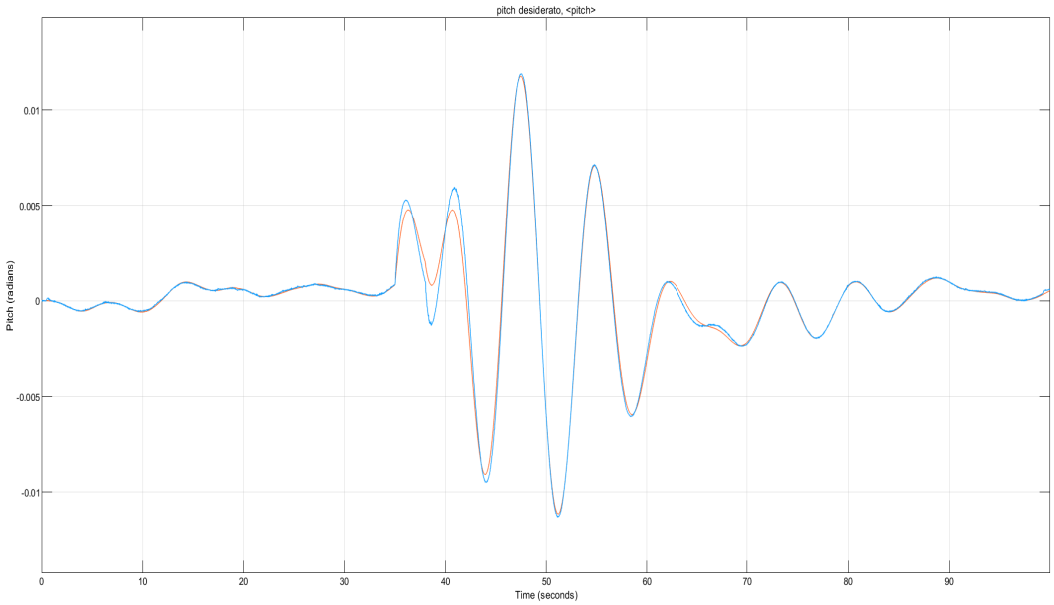


Figura 7.4: Risultato prima simulazione angolo di *pitch*

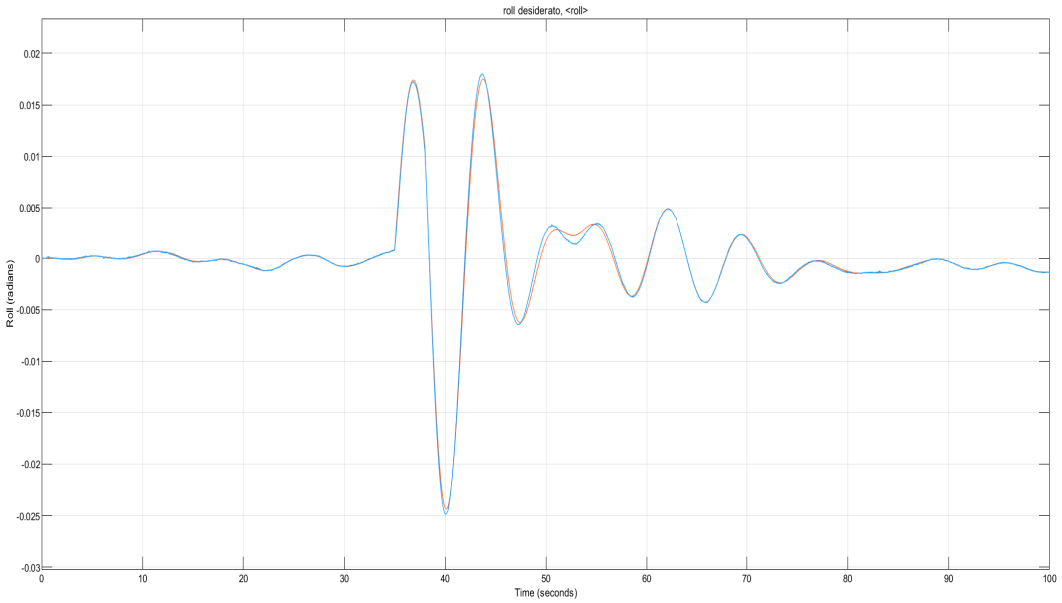


Figura 7.5: Risultato prima simulazione angolo di *roll*

7.2.1 Valutazione delle prestazioni

Per la valutazione delle prestazioni dell'algoritmo di controllo sliding mode del capitolo 5 è stato utilizzato l'indice **IAE:Integral Absolute Error**. Come si evince dall'acronimo è un indice che è calcolato integrando nel tempo (per tutta la durata di una simulazione) il valore assoluto dell'errore (differenza tra segnale di riferimento ed uscita del sistema). Nel caso in esame, lavorando a tempo discreto, l'integrale è stato convertito in una sommatoria discreta:

$$IAE = \sum_{k=0}^{T} \left| e(k) \right|$$

Nella figura 7.6 viene mostrato il calcolo dell'IAE per l'altitudine ma analogo è lo schema per il calcolo dell'IAE per gli angoli di roll, pitch e yaw.

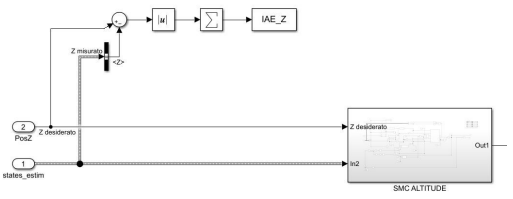


Figura 7.6: Calcolo dell'IAE per l'altitudine

Nella tabella 7.6 vengono dunque mostrati i valori dell'IAE associati all'altitudine ed agli angoli di roll, pitch e yaw.

Tabella 7.6: Valore dell'IAE (simulazione della durata di 100 secondi)

Variabile	IAE
Altitudine	$0,6934$
Yaw	$0,5207$
Pitch	$0,0162$
Roll	$0,0175$

7.3 Introduzione del disturbo *d*

Nella presente sezione vengono mostrate le simulazioni (della durata di 10 secondi) al calcolatore dell'algoritmo di controllo sliding mode non robusto, al quale è stato aggiunto il disturbo d_i introdotto nel capitolo 6:

$$d_i = \rho_i |S_i|^{p_i} |U_i|^{v_i} + d_{0i}$$

.

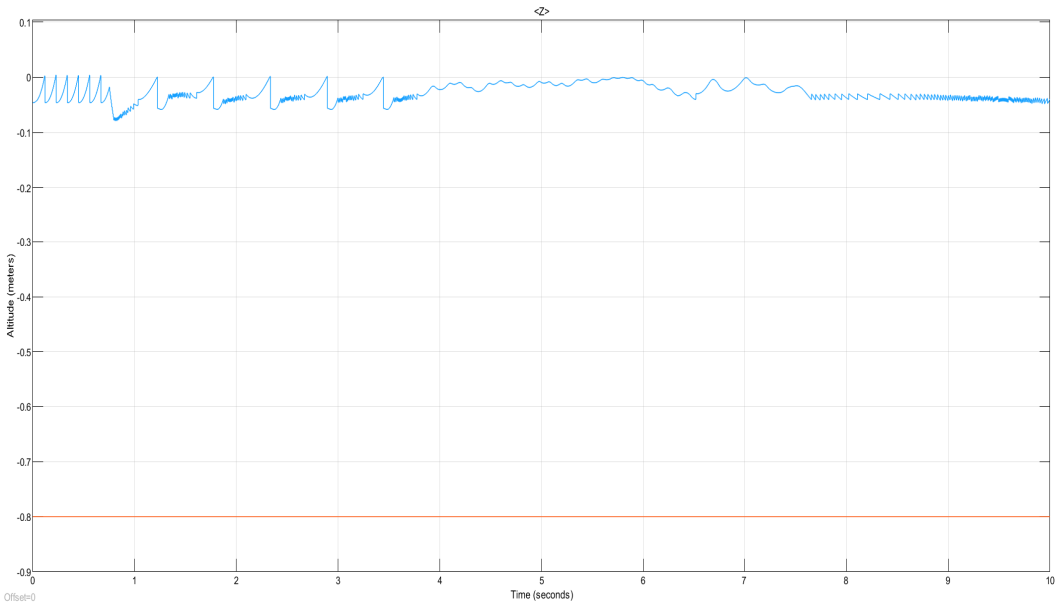


Figura 7.7: Disturbo d_i applicato sull'altitudine

Come è possibile notare dall'analisi delle figure 7.7, 7.8, 7.9 e 7.10, il controllore non robusto non è in grado di far fronte all'introduzione del disturbo d_i: per quanto riguarda l'asse z (altitudine) il sistema non è in grado di sollevarsi da terra, per quanto riguarda la posizione angolare (roll, pitch, yaw) si nota che il sistema diverge o ha comportamenti anomali e molto distanti dai riferimenti imposti.

7.4 Controllore sliding mode robusto

Nella presente sezione vengono mostrati i risultati delle simulazioni al calcolatore (della durata di 10 secondi) del sistema sottoposto al disturbo d_i, questa volta con il controllore robusto in luogo di quello "classico".

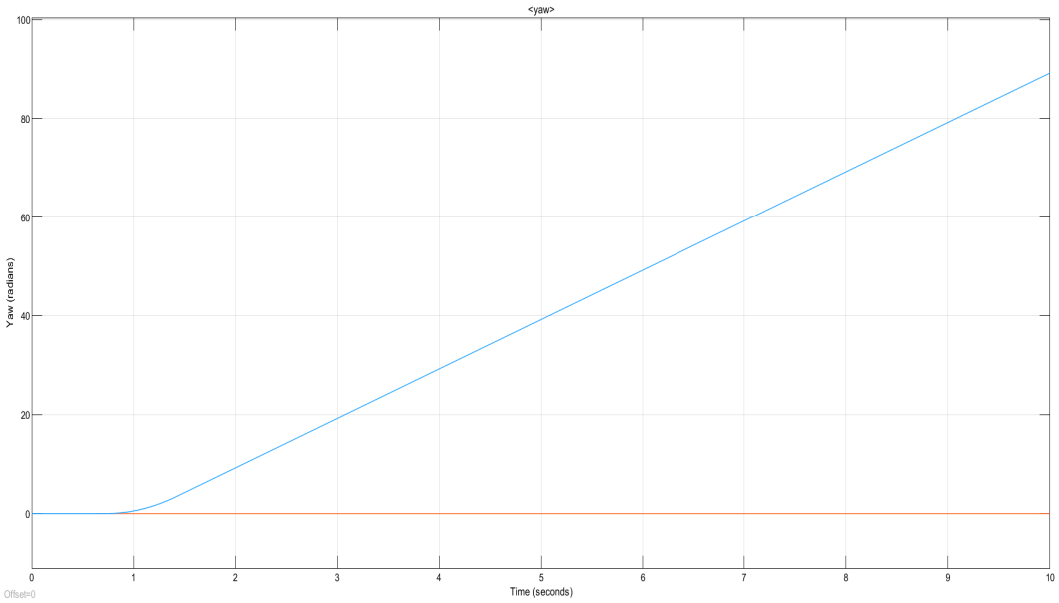


Figura 7.8: Disturbo d_i applicato sull'angolo di yaw

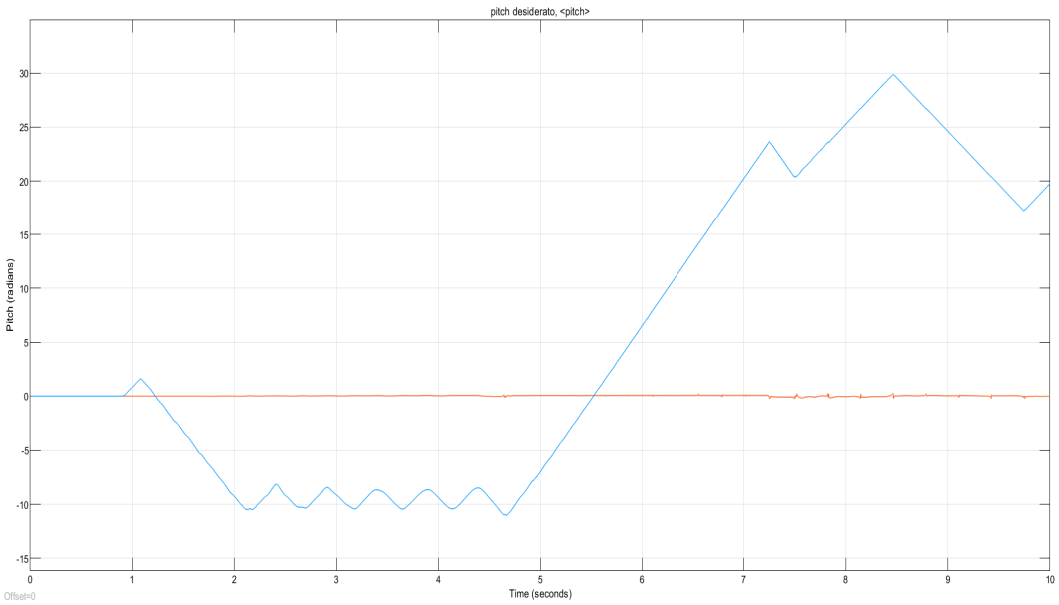


Figura 7.9: Disturbo d_i applicato sull'angolo di pitch

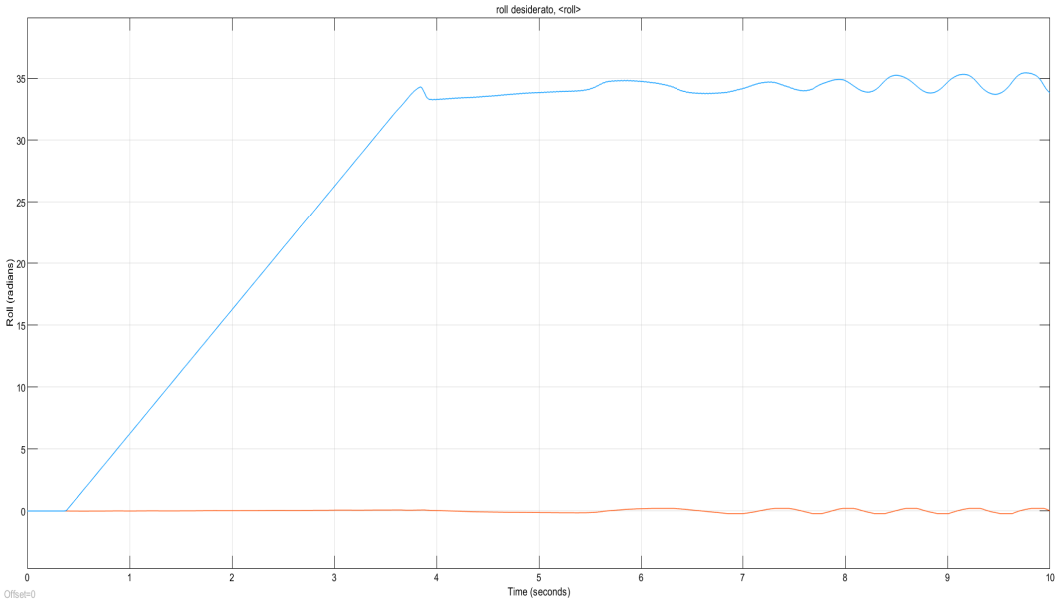


Figura 7.10: Disturbo d_i applicato sull'angolo di roll

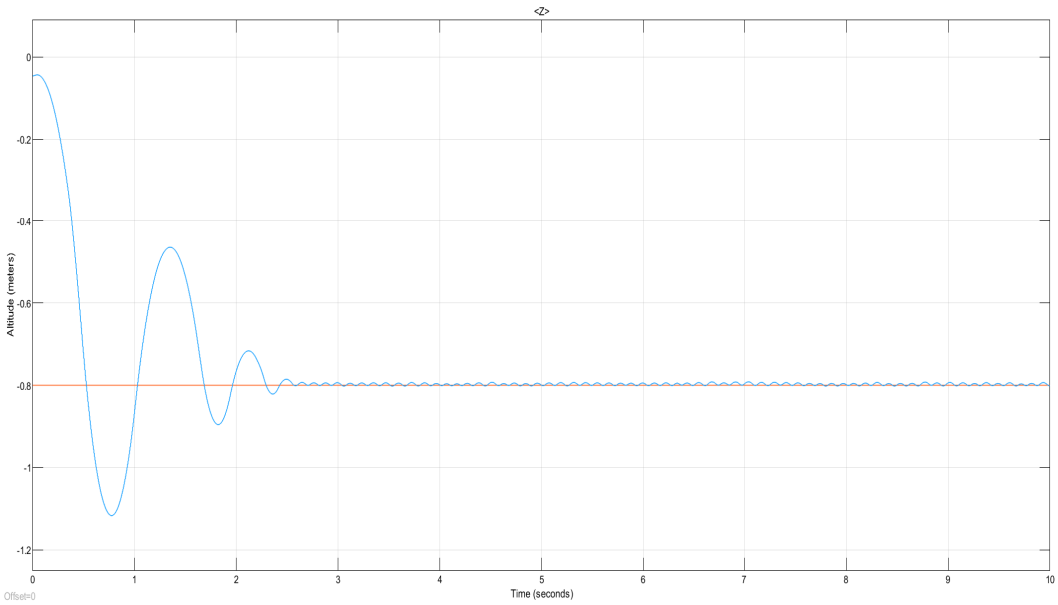


Figura 7.11: Disturbo d_i applicato sull'altitudine

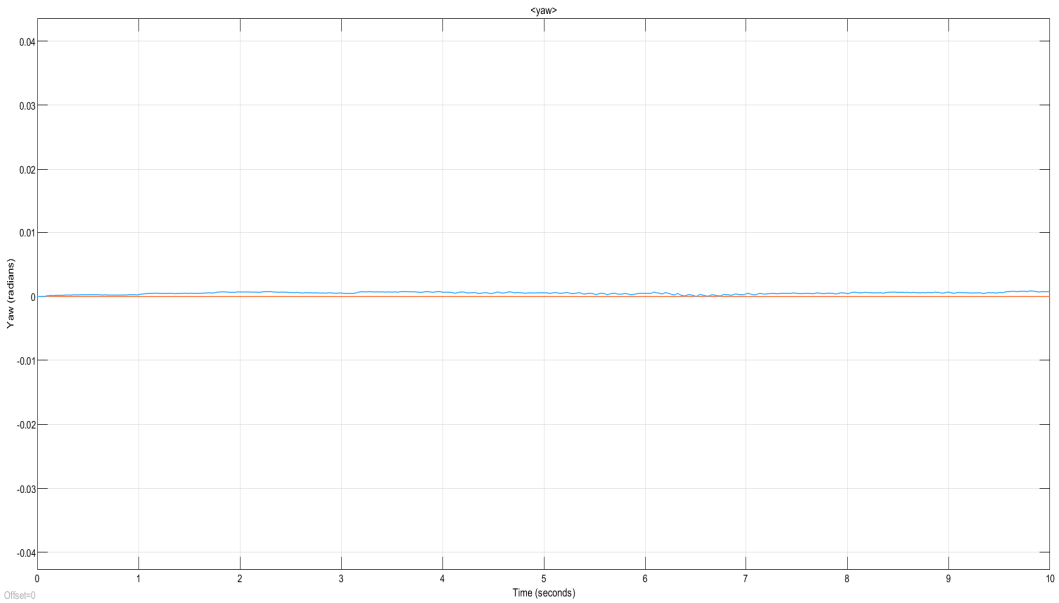


Figura 7.12: Disturbo d_i applicato sull'angolo di yaw

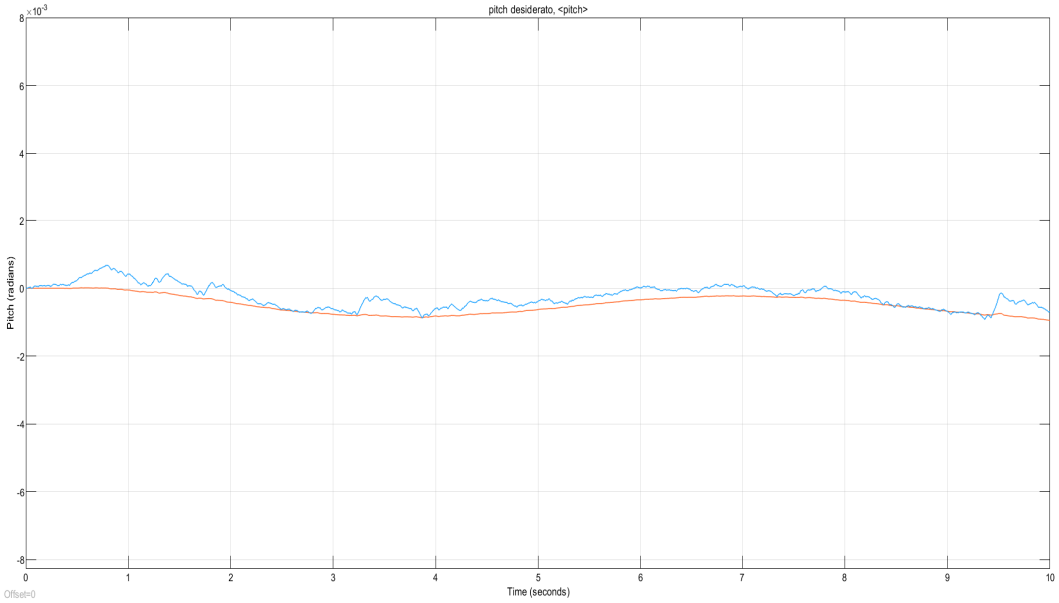


Figura 7.13: Disturbo d_i applicato sull'angolo di pitch

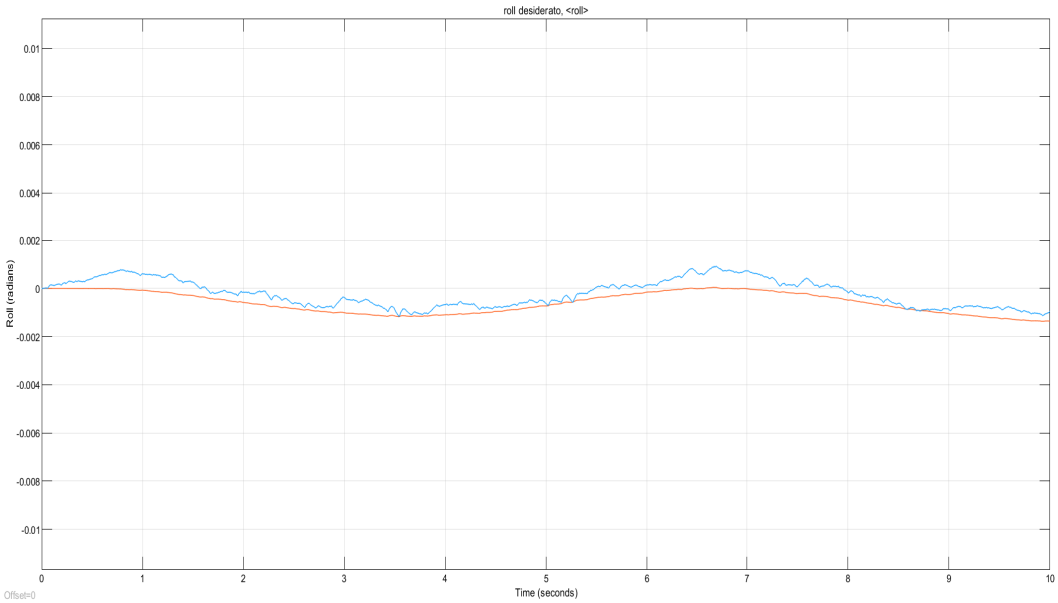


Figura 7.14: Disturbo d_i applicato sull'angolo di roll

Come è possibile notare dall'analisi delle figure 7.11, 7.12, 7.13 e 7.14, il controllore robusto proposto nella presente tesi è capace di compensare il termine positivo che è presente nella derivata della funzione di Lyapunov e che giustifica quindi la stabilità e la convergenza delle traiettorie del sistema.

7.5 Prove sperimentali sul sistema reale

A valle delle giustificazioni teoriche e dei risultati al calcolatore dell'algoritmo vengono mostrati, per completezza, alcune prove sperimentali sul sistema reale, ovvero sul drone *Parrot Mambo*.

7.5.1 Generazione del codice

In questa sezione viene mostrato come, attraverso l'interfaccia fornita dall'ambiente MATLAB & Simulink, sia possibile generare del codice C eseguibile dal drone a partire dalla programmazione grafica di blocchi Simulink. Si supponga di aver già implementato il controllore desiderato nel blocco Flight Control System (4.3) e di voler testare l'algoritmo di controllo sul sistema reale. Dopo aver acceso il drone *Parrot Mambo* ed averlo collegato al computer tramite bluetooth vanno seguiti i seguenti passaggi:

- Aprire la schermata MATLAB relativa al progetto asbQuadcopter, come mostrato in figura 7.15;
- Cliccare sull'icona, presente nella parte superiore dello schermo, chiamata *Generate flight code*;
- Attendere che il codice di volo venga generato. Alla conclusione delle operazioni, all'utente verranno proposte due finestre, rispettivamente mostrate nelle figure 7.16 e 7.17:

– Nella figura 7.16 viene mostrato il codice C che è stato generato, in modo tale da poter essere eventualmente revisionato e corretto;

– Nella figura 7.17 è presente l'interfaccia con la quale è possibile avviare il drone. In particolare è possibile fare quanto segue: avviare ed interrompere il volo del drone; decidere la durata del tempo in cui il drone è il volo; determinare, attraverso uno slider, la percentuale di potenza da erogare ai 4 motori del drone. Sarà quindi sufficiente cliccare sul pulsante verde *START* per autorizzare al volo il drone.

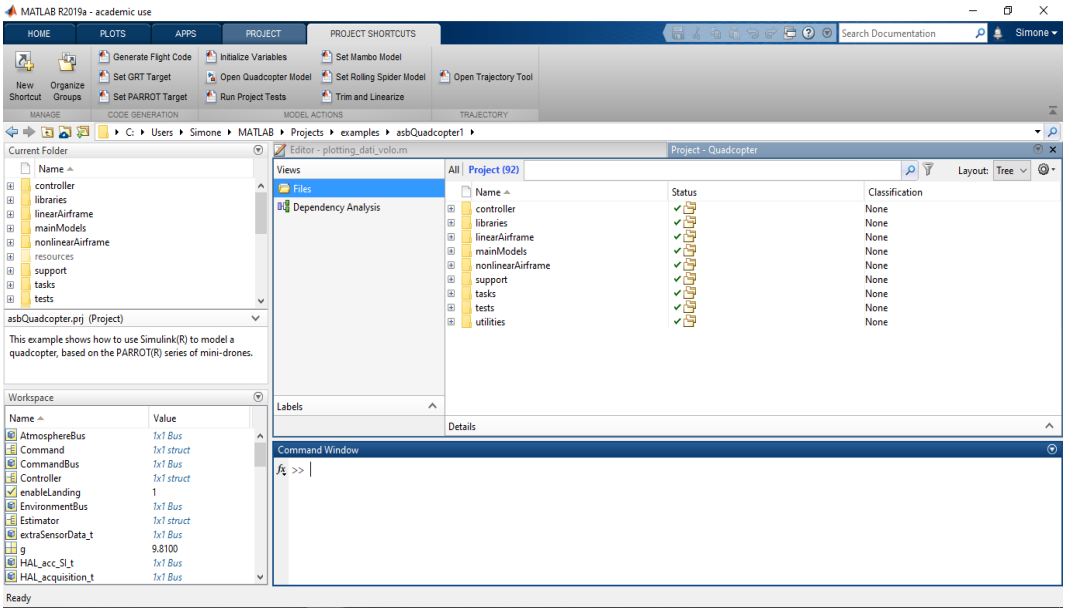


Figura 7.15: Interfaccia del progetto asbQuadcopter su MATLAB

7.5.2 Controllore sliding mode

In questa sezione vengono mostrati i risultati dell'implementazione del controllore sliding mode non robusto sul sistema reale. Nelle immagini 7.18 e 7.19 è possibile osservare il comportamento dell'algoritmo di controllo: sebbene il comportamento potrebbe discostarsi leggermente da quello delle simulazioni al calcolatore, bisogna ricordare come questo sia influenzato da molteplici fattori tra i quali disturbi ambientali, letture errate provenienti dai sensori, errori casuali e così via. In ogni caso si può affermare come anche sul sistema reale il comportamento dell'algoritmo (e di conseguenza del drone) siano soddisfacenti.

7.5.3 Introduzione di disturbi e confronto tra controllori sliding mode

In questa sezione verranno brevemente illustrati i risultati di ulteriori simulazioni svolte sul sistema reale, avendo questa volta impiegato il controllore robusto ed avendo sottoposto il sistema a disturbi.

- Per quanto riguarda l'asse z si è proceduto come segue: si è sfruttata la telecamera esterna (e non quella integrata nel fondo del drone che non è removibile) per emulare una variazione imprevista del peso del *Parrot Mambo*. Si è quindi voluto vedere se il controllore robusto riuscisse a far fronte a questa variazione nelle caratteristiche del

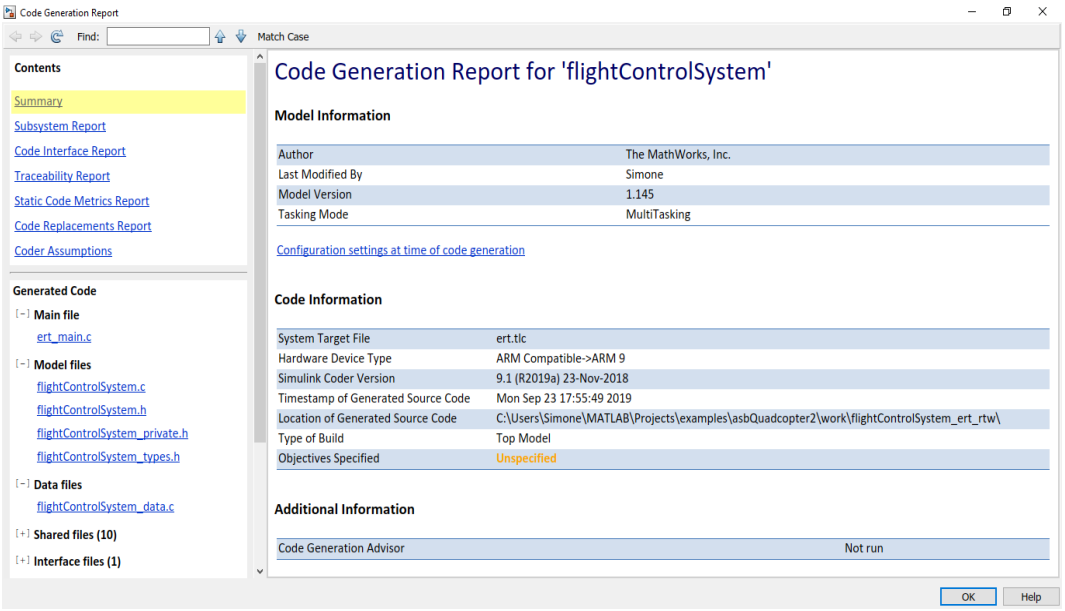


Figura 7.16: Report di generazione del codice

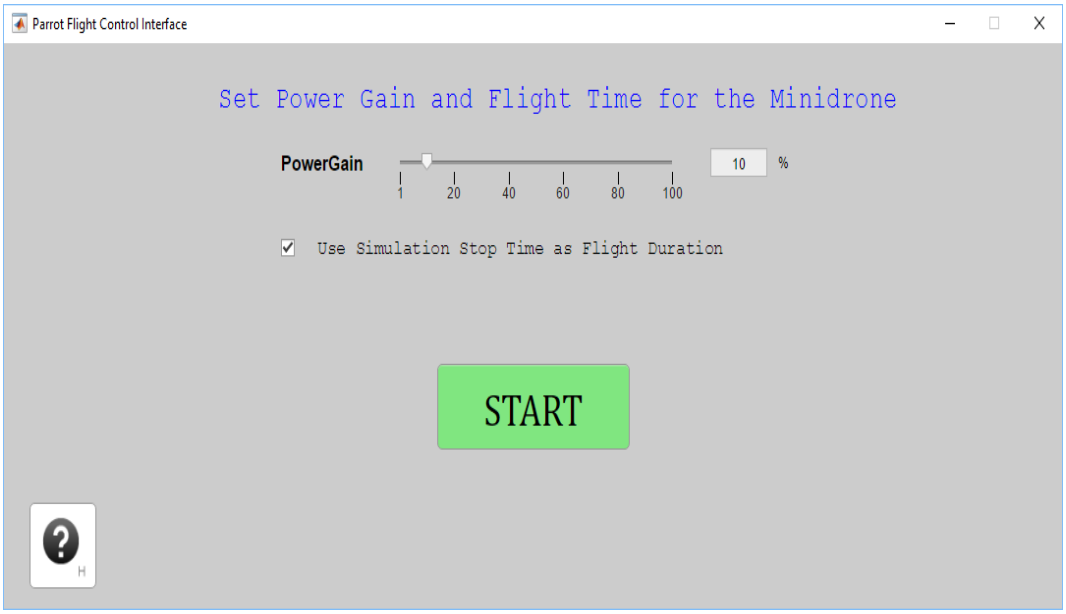


Figura 7.17: Flight Control Interface

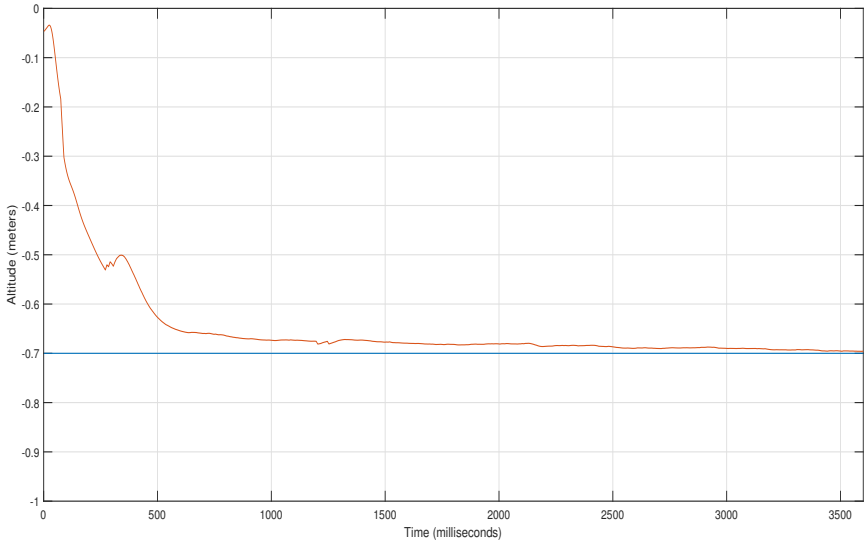


Figura 7.18: Controllore sliding mode per l'altitudine implementato su sistema reale

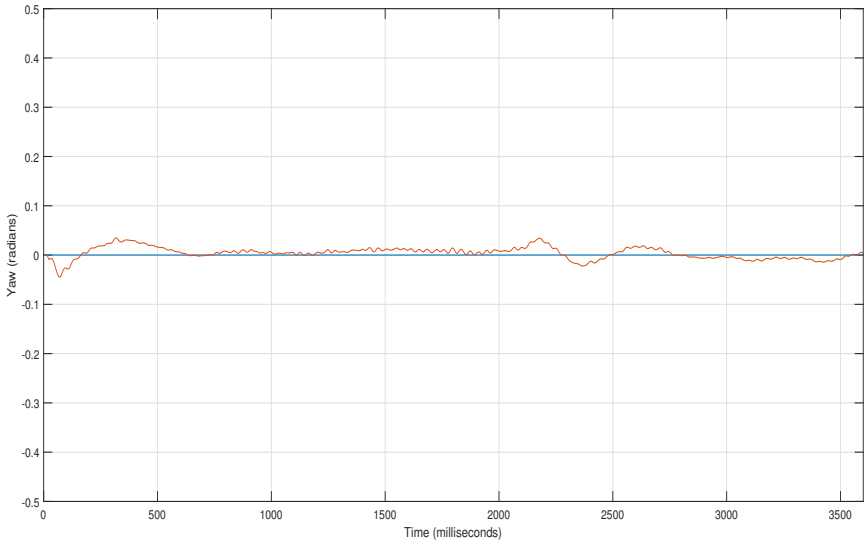


Figura 7.19: Controllore sliding mode per l'angolo di yaw implementato su sistema reale

sistema. La variazione di peso è inoltre considerevole: il drone pesa 63 grammi mentre la telecamera che è stata aggiunta ben 10. Si è dunque variato il peso di circa il 16%. Dal confronto delle figure 7.20 e 7.21 è possibile dedurre quanto segue:

  - il controllore non robusto non è in grado di far fronte al disturbo dovuto all'aggiunta di una massa: si nota infatti un offset (scostamento dal valore di riferimento) di circa 10 centimetri per tutta la durata della simulazione;
  - il controllore robusto è invece in grado di sopperire a questa variazione parametrica: nonostante la sovraelongazione iniziale riesce ad inseguire il riferimento a regime.

- Per quanto riguarda l'angolo di yaw si è proceduto come segue: è stato utilizzato un comune ventilatore per simulare dei disturbi aerodinamici (i.e. vento) ed analizzare la risposta dei due controllori, robusto e non, a tale disturbo. Quest'ultimo è stato "somministrato" al sistema per tutta la durata della simulazione e, per quanto possibile, si è cercato di effettuare entrambe le simulazioni nelle stesse condizioni operative. A conclusione di queste premesse va aggiunto come l'entità del disturbo aerodinamico sia stata esigua; ciò è dovuto dalla grande leggerezza del drone e dalla conseguente impossibilità di sottoporlo a disturbi di intensità maggiori .Dall'analisi delle figure 7.22 e 7.23 è possibile dedurre quanto segue:
  - il controllore non robusto è in grado di far fronte al disturbo ma con ampie variazioni rispetto al riferimento imposto. Il picco massimo di variazione è infatti di quasi 0,4 radianti, circa 20°;
  - anche il controllore robusto è in grado di far fronte al disturbo, questa volta con oscillazioni molto meno ampie rispetto al valore di riferimento e soprattutto sempre inferiori, in modulo, a 0,1 radianti, circa 5°.

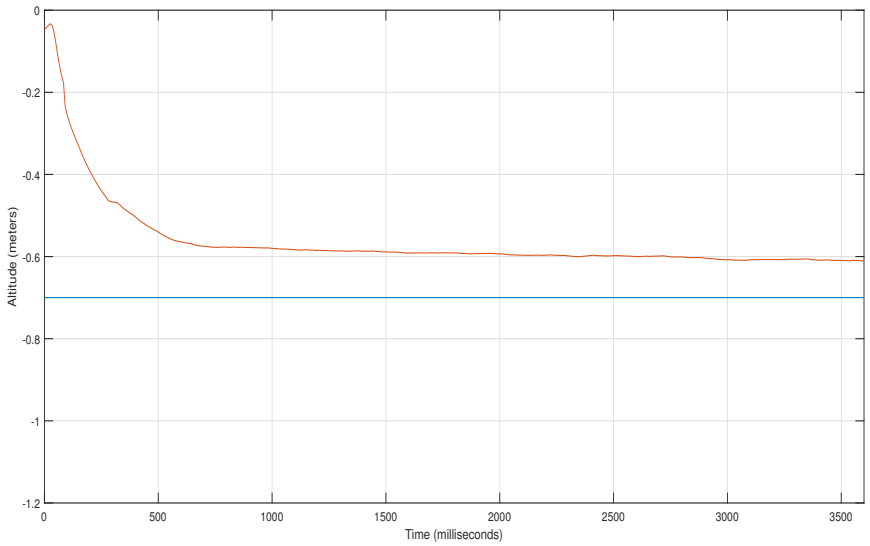


Figura 7.20: Risultato dell'aggiunta di una massa esterna al sistema con controllore non robusto

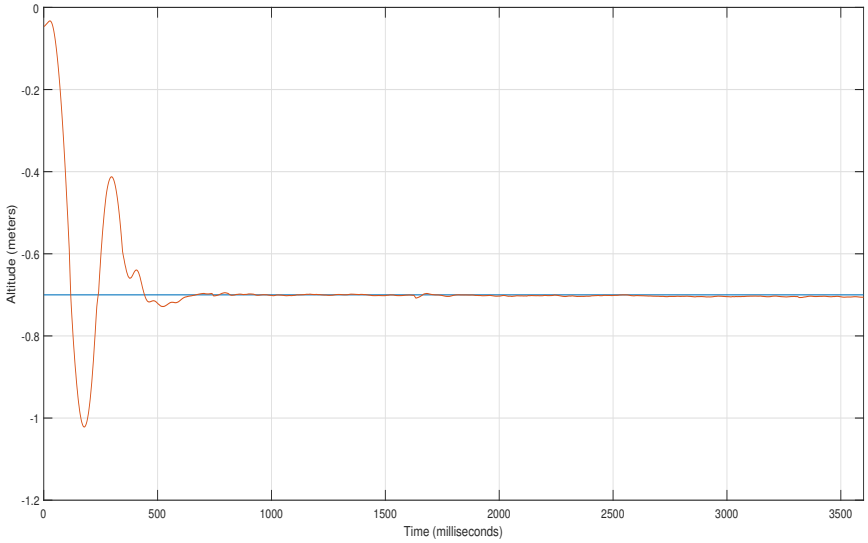


Figura 7.21: Risultato dell'aggiunta di una massa esterna al sistema con controllore robusto

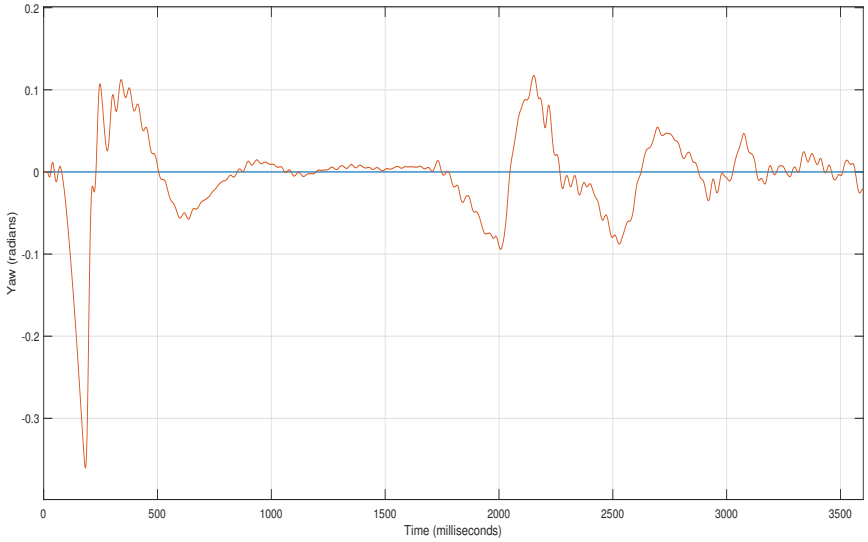


Figura 7.22: Risultato dell'introduzione di un disturbo esterno sullo yaw con controllore non robusto

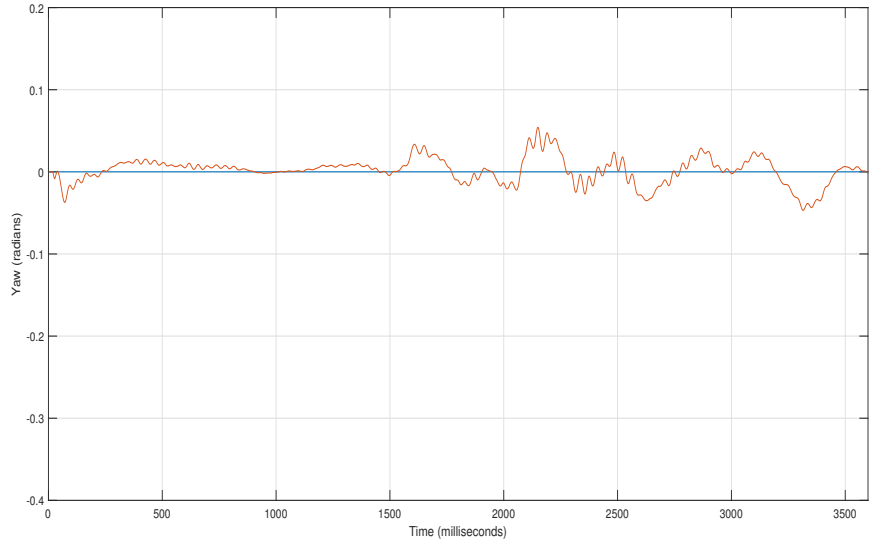


Figura 7.23: Risultato dell'introduzione di un disturbo esterno sullo yaw con controllore robusto

Capitolo 8

Conclusioni e sviluppi futuri

8.1 Conclusioni

Ricapitolando, nel presente lavoro di tesi è stato svolto quanto segue:

- si è scelto il sistema di interesse, in questo caso il drone *Parrot Mambo*;
- il drone è stato modellato matematicamente ed il suo comportamento sintetizzato attraverso le sue equazioni caratteristiche;
- si è studiato e conosciuto l'ambiente di sviluppo software attraverso il quale si sarebbero raggiunti gli obiettivi della presente tesi;
- si è studiato e sviluppato un controllore non lineare robusto che è andato a sostituire il controllore di volo di tipo PID che era presente originariamente sul drone;
- sono state svolti numerosi test al calcolatore e prove empiriche sul sistema reale per accertarsi dell'efficacia degli algoritmi implementati;
- le prove sperimentali di cui sopra sono state infine valutate.

Alla luce di quanto svolto si può affermare quanto segue:

- il modello scelto per la modellazione matematica del drone, noto in letteratura, si è rivelato efficace ai fini degli obiettivi di controllo desiderati;
- l'ambiente di sviluppo scelto, ovvero la *suite* MATLAB & Simulink, si è dimostrato, dopo un approfondito studio iniziale, adeguato (sebbene non intuitivo e scarsamente documentato) per studiare il comportamento del drone e per implementare, quasi senza sforzi progettuali nel passaggio dal calcolatore al sistema reale, gli algoritmi di controllo sul drone;
- il primo algoritmo di controllo sviluppato, ovvero il controllore sliding mode non robusto, ha dimostrato buone *performance* (calcolate tramite l'indice IAE - Integral Absolute Error), rispetto al controllore PID "tradizionale" che era inizialmente presente sul drone;
- il controllore del punto precedente si è rivelato però incapace di far fronte al disturbo successivamente introdotto, che rappresentava forze aerodinamiche non modellate nel sistema;
- per sopperire a tale mancanza è stato studiato e sviluppato un controllore robusto che si è dimostrato efficace nel risolvere il problema di cui sopra: le prestazioni del sistema con l'algoritmo di controllo robusto sono state testate sia al calcolatore che sul sistema

reale tramite aumenti della massa del sistema e perturbazioni esterne appositamente generate. In entrambi i casi si è potuta verificare l'utilità e l'adeguatezza del controllore robusto sviluppato nella risoluzione di queste problematiche di controllo.

In ogni caso, il lavoro svolto nella presente tesi getta le basi per ulteriori studi ed approfondimenti sia per quanto riguarda lo sviluppo di altre tipologie di sistemi di controllo ed il confronto tra queste ultime, sia per quanto riguarda l'architettura Simulink, che valorizzerebbe e migliorerebbe ulteriormente gli schemi e le tecniche di controllo di cui sopra.

8.2 Sviluppi futuri

Da quanto esposto nella presente tesi e dalle conclusioni a cui si è giunti vengono di seguito elencati i possibili sviluppi futuri:

- sviluppare un software euristico per il *tuning* automatico dei parametri di progetto, in quanto nel presente lavoro la scelta di tali parametri è stata svolta manualmente e verificando di volta in volta l'andamento delle prestazioni del sistema di controllo;
- includere nella modellazione matematica del sistema anche la fotocamera verticale: l'informazione sulla stima della velocità proveniente dalla fotocamera verticale non è stato utilizzato nel presente lavoro; sarebbe quindi opportuno investigare eventuali miglioramenti prestazionali includendo nel *loop* di controllo anche i dati provenienti dal suddetto sensore;
- implementare gli algoritmi di controllo *sliding mode* non solo tramite l'ambiente MATLAB & Simulink ma anche utilizzando altri linguaggi di programmazione ad alto livello come javascript e python (supportati dalla piattaforma del Parrot Mambo) e fare una comparazione dei seguenti aspetti:
  - complessità computazionale;
  - fedeltà tra il comportamento in simulazione e quello sul sistema reale;
  - difficoltà di implementazione degli algoritmi stessi;
  - *performance* degli algoritmi.
- implementare, tramite l'ambiente MATLAB & Simulink, altre tecniche di controllo non lineari (*model-based*) e confrontarne le prestazioni;
- modificare, all'interno dello schema Simulink, anche gli stimatori dello stato, verificare se ve ne siano di più opportuni e di più adatti e confrontarne in seguito le prestazioni.

Bibliografia

[1] Alfredo Roma. Breve storia dei droni. http://www.limesonline.com/breve-storia-dei-droni/48678.

[2] Mario Mingoia. Applicazioni droni. https://www.abcdroni.it/applicazioni-droni.

[3] Parrot. Parrot Educational website. https://edu.parrot.com.

[4] Alessandro Freddi. Slide del corso di Misure e Strumentazione per l'automazione.

[5] Fum Wei Zhong. Implementation of Simulink controller design on Iris+ quadrotor. Naval Postgraduate School Thesis, Monterey, California.

[6] Li Z. Murray, R. M. and S. S. Sastry. *A mathematical introduction to robotic manipulation*, volume page 167. CRC Press, 1994.

[7] A. Azzam and X. Wang. Quadrotor aerial robot dynamic modeling and configuration stabilization. *Control, Automation and Robotics (CAR), 2nd International Asia Conference*, 1:438–444, 2010.

[8] Bresciani, T. Modelling, identification and control of a quadrotor Helicopter. masters thesis, Lund university.

[9] Wikipedia. Pagina Matlab. https://it.wikipedia.org/wiki/MATLAB.

[10] Mathworks. Parrot Minidrones Support from Simulink. https://it.mathworks.com/hardware-support/parrot-minidrones.html.

[11] Mathworks. Pagina ufficiale Matlab. https://it.mathworks.com/products/matlab.html.

[12] Alberto Isidori. *Sistemi di controllo.* Edizioni Siderea, 1993.

[13] Hassan K. Khalil. *Nonlinear Systems.* Prentice Hall, 1996.

[14] Singh, G. K. and Hol e, K. E. Guaranteed performance in reaching mode of sliding mode controlled systems. Sadhana, 29(1): 129141.

[15] Khalid Khan M. Design and application of second order sliding mode control algorithms. Ph.D. thesis, university of Leicester.

[16] Denis Efimov Maan El Badaoui El Najjar Boussad Abci, Gang Zheng. Robust altitude and attitude sliding mode controllers for quadrotors. *IFAC PapersOnLine 50-1*, 1:2720–2725, 2017.

[17] Politecnico di Torino. Reiezione di disturbi in regime permanente. http://corsiadistanza.polito.it/on-line/Controlli_automatici/pdf/U3_2.pdf.

[18] Prof. Giuseppe Fusco. Complementi di controlli Automatici. http://www.docente.unicas.it/useruploads/000371/files/stabilità_robusta.pdf.