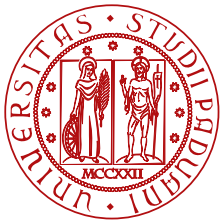


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA AEROSPAZIALE

TESI DI LAUREA MAGISTRALE

Progetto preliminare e studio delle prestazioni di velivoli multirotore: strumenti di analisi

Relatore
Prof Emanuele Luigi de Angelis

Laureanda
Alice Moccia Marotto
matricola 2061477

ANNO ACCADEMICO 2023-2024

19 APRILE

Anyway
nothing is simple

Ringraziamenti

Sommario

I velivoli a pilotaggio remoto riscuotono un interesse sempre maggiore nel mondo scientifico soprattutto perché rappresentano la soluzione ottimale per un gran numero di applicazioni differenti. Tra queste possiamo ricordare il trasporto di carichi, la ricerca e il soccorso, la gestione del rischio, la sorveglianza, l'aero-fotogrammetria e, in generale, le attività di rilevamento remoto. Tra le varie configurazioni disponibili i multirotori rappresentano l'alternativa maggiormente implementata. Ricordiamo infatti che la bassa complessità strutturale e la semplicità d'uso consentono una riduzione dei costi operativi. Dall'altro lato, le capacità di stazionamento e di decollo e atterraggio verticali permettono una migliore capacità di manovra. Lo studio di tale argomento mediante un approccio ingegneristico è di cruciale importanza per differenti aspetti. Infatti, si ha la necessità progettare i diversi sistemi presi in considerazione e ottimizzarne le prestazioni a seconda degli obiettivi che si vogliono ottenere. Per tale scopo viene quindi implementata un'interfaccia grafica mediante l'utilizzo di App Designer di MATLAB. Tale strumento renderà possibile la creazione di un'applicazione interattiva con una grafica utente (GUI) utilizzando elementi visivi come pulsanti, grafici e caselle di testo. Questo consentirà di effettuare in modo intuitivo il progetto preliminare del multirotore e successivamente di studiarne le prestazioni per garantirne l'efficienza. In questo modo quindi si otterrà un ausilio per lo studio di carattere ingegneristico del velivolo a pilotaggio remoto.

Abstract

Remotely–Piloted Aerial Systems are gaining increasing interest in the scientific world, especially because they represent the optimal solution for a large number of applications. Among these we can remember the load transportation, search and rescue, risk management, surveillance, aero–photogrammetry, and, in general, remote sensing activities. Among the various configurations available, multirotors represent the most popular alternative implemented. In fact, the low structural complexity and simplicity of use allow a reduction in operating costs. On the other hand, their hovering capabilities and vertical take-off and landing enhance maneuverability. The study of this topic through an engineering approach is crucial for various reasons. There is a need to design the different systems under consideration and optimize their performance according to the desired objectives. In the context of this thesis work, a graphical interface is designed and programmed using MATLAB's App Designer. This interface will allow for an intuitive preliminary design of a multirotor and, subsequently, a detailed study of its performance. In this way, it will serve as a tool for the engineering study of remotely–piloted aerial systems .

Indice

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni l'interesse nei confronti di sistemi a pilotaggio remoto è notevolmente aumentato. [1] Questo è da imputarsi ai vantaggi che l'uso di tali tecnologie comporta e le varie funzioni che si possono svolgere.

Analizzando le caratteristiche strutturali è possibile notare che tale tecnologia, confrontata con le altre in commercio, presenta una maggiore semplicità strutturale ed impiantistica. Questo porta così ad avere una riduzione dei costi operazionali in quanto essi sono dipendenti dalla massa totale del velivolo. In secondo luogo, si può osservare un'alta facilità di utilizzo. Analizzando le capacità di volo infatti è possibile notare che i multirotori offrono la possibilità di effettuare un decollo e un atterraggio verticale. In questo modo possono operare in aree ristrette o comunemente non adatte al volo. Per queste caratteristiche vengono spesso paragonate agli elicotteri tradizionali ma, a parità di peso, i sistemi a pilotaggio remoto possiedono dimensioni inferiori e maggiore manovrabilità. [2] [3] [4] [5]
Tali qualità hanno portato alla possibilità di ampliare il range di applicazioni di interesse. Possiamo infatti notare che i sistemi a pilotaggio remoto (Remotely-Piloted Aerial System RPAS) possono essere impiegati in differenti ambiti come ad esempio: ricognizione e sorveglianza, agricoltura, industria petrolifera e del gas, trasporto di carichi, monitoraggio ambientale, ricerca e soccorso, ricerca scientifica, sicurezza pubblica, aerofotogrammetria e molto altro. [6] [7]
Ciò che accomuna tutte queste applicazioni è la necessità di operare in ambienti che potrebbero essere considerati dannosi per l'uomo o in cui comunemente non possono essere utilizzate altre tecnologie, in quanto la morfologia del territorio e le condizioni ambientali non ne rendono possibile il volo.

Si può quindi dedurre che la progettazione dei velivoli a pilotaggio remoto debba essere dettagliata e differenziarsi in base alla diversa applicazione a cui si deve rispondere. Questo comporta la necessità di avere specifiche e requisiti differenti per il singolo caso preso in esame, così da rendere la tecnologia idonea allo scopo per la quale deve essere impiegata.
Recentemente, quindi, è nata la necessità di affrontare tale problema utilizzando un approccio ingegneristico. Per gestire e agevolare l'uso sempre più diffuso di multirotori, infatti, si deve implementare un metodo per stimare le prestazioni e ottenere un dimensionamento preliminare. Per affrontare tale studio si è scelto di implementare un metodo basato sulle frazioni di peso, di tipico impiego aeronautico. [8]
La base di tale approccio è quella di considerare i componenti che costituiscono il multirotore singolarmente e studiarne le caratteristiche. Successivamente verrà considerato l'insieme delle singole

unità e di conseguenza le caratteristiche complessive della tecnologia.

In secondo luogo, essere a conoscenza delle caratteristiche operative consente anche di poter confrontare differenti velivoli e determinare a priori la scelta più efficiente per raggiungere lo scopo desiderato.
Questo comporta un vantaggio sia dal punto di vista economico sia per quanto riguarda il tempo di progettazione.
In precedenza, infatti, per assemblare e rendere operativo un multirotore, si doveva procedere per tentativi. Questo implicava l'acquisto dei vari componenti, ipotizzati adeguati, l'assemblaggio degli stessi e il successivo studio di prestazioni. Non si poteva quindi avere a priori una stima delle proprietà e quindi del lavoro che avrebbe potuto svolgere tale tecnologia e inoltre si aveva la necessità di modificare la tipologia di unità implementate per ottenere i risultati ricercati.

Considerando il processo per determinare le prestazioni di un velivolo a pilotaggio remoto (RPAS), si nota che si ha la necessità di fare riferimento ad una serie di fasi, spaziando dall'analisi dei requisiti operativi alla gestione dell'energia.
In seguito, verrà fatto un accenno dei differenti elementi da analizzare. Questi rappresentano le fasi successive che devono essere svolte in fase di progettazione: definizione dei requisiti operativi, stima della massa e posizione del centro di gravità, analisi aerodinamica, esame del sistema propulsivo, analisi della dinamica di volo, gestione dell'energia, valutazione di autonomia e carico utile, analisi delle prestazioni in diverse condizioni, simulazioni e verifiche sperimentali, ottimizzazione del design.
Da ciò è possibile quindi notare che il calcolo delle prestazioni di un RPAS rappresenta un processo complesso e multidisciplinare che coinvolge diverse discipline dell'ingegneria aerospaziale. Si ricorda che l'approccio specifico per la progettazione del singolo multirotore dipende dalle caratteristiche del velivolo e dagli obiettivi operativi desiderati.
Per ottenere la conoscenza di una tecnologia che si vuole sviluppare, può essere considerare sufficiente operare una stima ingegneristica delle prestazioni dei vari sottosistemi che la compongono. Questo ha lo scopo di ridurre i costi operazionali e rendere di più facile esecuzione lo studio così da ottenere risultati che rispecchiano sufficientemente la realtà diminuendo i tempi di esecuzione. Per ottenere ciò si possono usare due differenti approcci.

Il primo consiste in un approccio puramente matematico. Servendosi di funzioni analitiche, si ha la possibilità di ottenere valori stimati delle caratteristiche del velivolo. In questo caso si deve ricordare che, per arrivare al risultato cercato, si è reso necessario operare semplificazioni o utilizzare modelli matematici che non descrivono perfettamente la realtà ma la rappresentano in modo sufficientemente preciso. [9]

Il secondo approccio, su cui verterà il lavoro di tesi, consiste nell'analisi basata su dati sperimantali. Si dovrà infatti fare riferimento a test effettuati su specifici impianti propulsivi, così da ottenerne i valori delle prestazioni. Questi valori sono tabulati in database pubblici creati col fine di rendere disponibili più conoscenze possibili relative a differenti tecnologie presenti sul mercato. [10]
Per affrontare tale studio si è ricorsi all'implementazione di un software in grado di acquisire i valori tabulati e, successivamente, elaborarli per il calcolo delle incognite ricercate. Per rendere possibile l'utilizzo dell'applicazione si dovrà associare alla tecnologia scelta per l'unità propulsiva quella relativa alla batteria da implementare, ricordando che andrà ad influenzare in modo non

trascurabile il peso del velivolo e quindi, conseguentemente, le sue prestazioni.
In ultimo luogo si deve ricordare che anche le relative condizioni operative, come ad esempio la quota di volo, influenzeranno il rendimento del sistema. L'uso di tale applicazione permetterà di rendere note le prestazioni, come ad esempio il tempo di scarica o la potenza elettrica necessaria del velivolo preso in esame.
Successivamente, si avrà anche la possibilità di confrontare tale sistema con un altro in cui verranno variati uno o più parametri, così da poterne effettuare un confronto e stabilire quale tecnologia sia più efficiente per l'applicazione che il RPAS deve svolgere.

Per validare tale lavoro di tesi e quindi mettere in luce che i dati ottenuti mediante l'uso dell'applicazione siano affidabili, verranno riportati tre differenti esempi. Questi prenderanno in considerazione tre tipologie di velivolo già studiate attraverso l'approccio matematico e tali risultati verranno messi a confronto con quelli ottenuti tramite l'applicazione implementata, così da stabilire quanto essi approssimano i risultati teorici e, in caso contrario, quali siano i limiti di tale metodo sperimentale.

Capitolo 2

Modellazione matematica

La necessità si predire l'autonomia e la dimensione preliminare ottimale di un multirotore ha portato alla determinazione di funzioni analitiche specifiche per ottenere tali risultati.

2.1 Volo a punto fisso (hover)

In primo luogo, verrà analizzata solo la condizione di volo a punto fisso. Questa è una condizione per cui il multirotore rimane stazionario rispetto ad un punto specifico sulla superficie terrestre. Il volo a punto fisso viene spesso utilizzato in applicazioni come la fotografia aerea e il monitoraggio ovvero tutte quelle applicazioni in cui è necessario mantenere una vista costante su un determinato punto fisso.

Le forze principali agenti sul velivolo in hover sono due: il peso (W) e la trazione (T).
La trazione è generata dai rotori e ha lo scopo di contrastare la forza peso per garantire il volo a punto fisso.

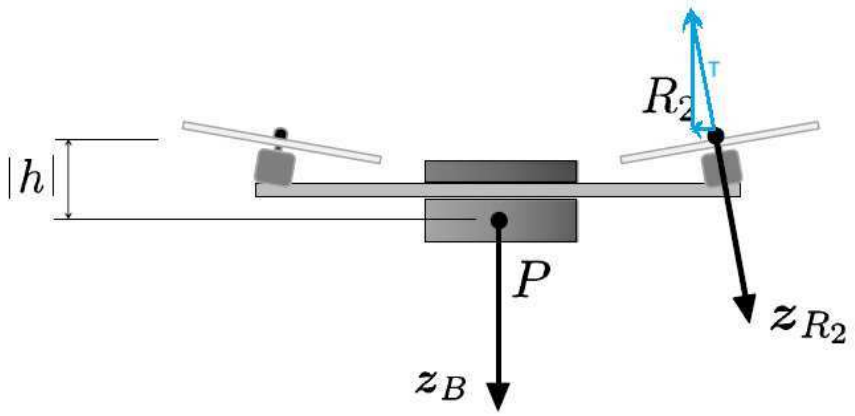


Figura 2.1: riduzione della trazione dovuta all'inclinazione dell'elica

Per permettere il volo la somma delle forze verticali deve essere nulla. Nell'analisi di tali forze si deve considerare anche l'inclinazione che possono avere i singoli rotori rispetto al piano orizzontale. Dalla figura (2.1) si può osservare che la trazione generata dall'elica non sarà in generale perpendicolare al piano orizzontale. A tal proposito si definiscono i seguenti angoli:

- Angolo di diedro (Γ): è l'angolo che si ottiene ruotando il disco del rotore in modo che una componente della spinta T_j sia diretta lungo il braccio del rotore j-esimo. Questo angolo

è responsabile della capacità del multirotore di inclinarsi in avanti, indietro o lateralmente dando così la possibilità al velivolo di muoversi in diverse direzioni. È considerato positivo quando la spinta del rotore è orientata nella direzione del baricentro del veicolo.

- Angolo di tilt (ξ): è l'angolo che consente di generare una componente della spinta che è ortogonale al piano verticale locale che contiene il braccio del rotore j-esimo. Questo angolo si presume positivo quando la componente relativa della forza esercita un momento positivo intorno a z_B. La sua importanza risiede nel fatto che esso opera per migliorare la stabilità in imbardata del velivolo.

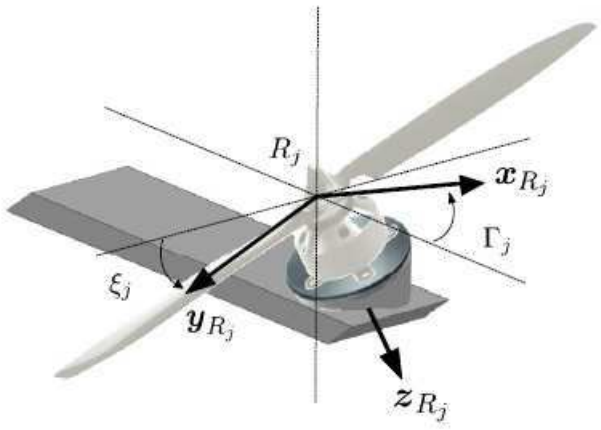


Figura 2.2: descrizione angoli di tilt e diedro

Di conseguenza, possiamo determinare la spinta generata dal singolo rotore considerando la seguente equazione di equilibrio:

$$T = \frac{W}{N\cos(\Gamma)\cdot\cos(\xi)} \tag{2.1}$$

dove il valore N è il numero di rotori che sono implementati sul velivolo da analizzare.

2.2 Considerazioni aeromeccaniche

Prendiamo in esame un multirotore in hover dotato di N unità propulsive identiche.
La potenza totale necessaria per il volo può essere espressa attraverso la seguente funzione analitica:

$$P_{\mathrm{r}} = P_{\mathrm{s}} + P_{\mathrm{h}} \tag{2.2}$$

ovvero la somma tra la potenza assorbita dai sistemi di bordo (P_s), che può essere considerata costante in questa analisi, e quella necessaria per attuare il volo stazionario (P_h).
Quest'ultima può essere determinata a partire dalla conoscenza della potenza indotta ideale del singolo motore elettrico (P_{id}) e della figura di merito (f):

$$P_{\mathrm{h}} = \frac{NP_{\mathrm{id}}}{f} \tag{2.3}$$

Si può ora considerare che W=mg sia il peso al decollo e P_{id}= T vi dalla teoria del momento, considerando una velocità vi uniforme sul disco attuatore.

Seguendo l'ipotesi di Glauert la velocità indotta si può esprimere come:

$$vi = \sqrt{\frac{T}{2\rho A}} \tag{2.4}$$

dove ρ è la densità dell'aria e A rappresenta l'area del disco del rotore ($\omega\ R^2$) dove R è il raggio del disco. Per la semplificazione dei calcoli si può trascurare l'effetto che la velocità indotta ha sulla struttura in quanto tale contributo può essere considerato trascurabile in fase di progetto preliminare.

Nell'analisi delle prestazioni del sistema è importante determinare come le perdite di potenza influenzino il valore di energia che la batteria deve rendere disponibile per garantire il volo.
Queste si possono ricondurre a perdite dovute alla lunghezza dei cavi per il trasporto dell'energia, ai regolatori di velocità elettronici (ESC) e alle perdite presenti nel motore elettrico adottato.
Ciascuno dei differenti sottosistemi possiede un rendimento proprio. Nella trattazione però verrà considerato il rendimento elettrico complessivo η_{e} dato dal prodotto dei singoli rendimenti. Sapendo però che le perdite relative ai cavi elettrici sono molto inferiori rispetto agli altri contributi, queste potranno essere trascurate in fase di progettazione preliminare.
I valori di rendimento possono essere forniti dai produttori di sistema, ricavati da database online oppure ricavati per via sperimentale.

È allora in questo modo possibile determinare la potenza necessaria per il volo $\mathrm{P_h}$:

$$P_{\mathrm{h}} = \eta_{\mathrm{e}}(P_{\mathrm{b}} - P_{\mathrm{s}}) \tag{2.5}$$

dove $\mathrm{P_b}$ è la potenza totale generata dalla batteria. Ricordando l'equazione (2.2) può essere riscritta come:

$$P_{\mathrm{b}} = P_{\mathrm{s}} + \frac{NP_{\mathrm{id}}}{f\eta_{\mathrm{e}}} \tag{2.6}$$

L'espressione mette in luce la necessità di conoscere il valore del coefficiente di merito per il calcolo della potenza. La caratterizzazione di tale grandezza si basa sulla conoscenza dettagliata dei coefficienti aerodinamici del sistema propulsivo, questo implica il dover eseguire misurazioni o calcoli complessi. In tale lavoro di tesi la figura di merito sarà considerata dipendente da pochi parametri dell'elica, derivanti da misurazioni semplici o dalle specifiche messe a disposizione dal costruttore. Per non inficiare la ricerca del valore di potenza cercato vengono presi in considerazione dei coefficienti di correzione in grado di predire al meglio le prestazioni di una classe di eliche standard ottimizzate.

Consideriamo la velocità dell'estremità delle pale $\mathrm{v_{tip}}= \Omega \mathrm{R}$ e il coefficiente di trazione $\mathrm{C_T}=\frac{T}{\rho A {V_{\mathrm{tip}}}^2}$.
La funzione che descrive il coefficiente di merito risulta essere:

$$f = \frac{\frac{{C_{\mathrm{T}}}^{3/2}}{\sqrt{2}}}{k_{\mathrm{ind}}\frac{{C_{\mathrm{T}}}^{3/2}}{\sqrt{2}} + \frac{\sigma C_{\mathrm{d}}}{8}} \tag{2.7}$$

dove σ è la solidità del rotore, $\mathrm{C_d}$ è il coefficiente medio di resistenza aerodinamica e $\mathrm{k_{ind}}$ è fattore di correzione della potenza indotta derivante dalla presenza di effetti non ideali.

Considernado ora il coefficiente di portanza $C_l = C_{l\alpha}(\theta - \alpha_0 - \Phi)$, dove $C_{l\alpha}$ è pendenza del coefficiente di portanza bidimensional, α_0 è l'angolo cui corrisponde portanza nulla, θ è l'angolo di beccheggio e Φ è l'angolo di flusso relativo in una sezione generica dell'ala dovuto al flusso indotto. Nello studio del fattore di merito, col fine di ridurre l'onere di calcolo, si assumono valide le seguenti assunzioni:

- k_{ind} ha fluttuazioni che comportano un impatto limitato nel calcolo di f e viene quindi considerato costante.
- Il coefficiente di resistenza aerodinamica viene assunto costante per rotori di grandi dimensioni ($C_d=C_{d0}$) mentre viene considerato dipendente dal numero di Reynolds nella trattazione di rotori di piccole dimensioni ($C_d=C_d(Re)$). [11]
- Il coefficiente di attrito segue la teoria di Blasius ed è espresso come $C_f = 1.328\sqrt{Re}$. Il coefficiente di resistenza aeroinamica può allora essere espresso come $C_d=2C_f$. [12] Questa espressione è in grado di descrivere in maniera opportuna gli effetti dati dal numero di Reynold sulle perdite di profilo.
- $C_{l\alpha}$ e α_0 sono considerati costanti lungo la pala.

Prendendo in considerazione l'angolo di beccheggio relativo al 75% della corda è possibile ottenere le seguenti funzioni per il coeffiiente di trazione:

$$C_T = \frac{1}{2}\sigma C_{l\alpha}\left(\frac{\theta_{75} - \alpha_0}{3} - \frac{\lambda}{2}\right) \tag{2.8}$$

dove λ è data dalla velocità indotta fratto la velocità all'estremità della pala.
L'equazione precedente può essere riscritta come:

$$C_T = 2\left(\frac{v_i}{v_{tip}}\right)^2 \tag{2.9}$$

Andando a considerare ora le equazioni (2.8) e (2.8) è possibile determinare un'espressione con incognita v_{tip}. Escludendo la soluzione di valore negativo, la velocità all'estremità della pala, ottenuta mediante BET, risulta essere:

$$v_{tip}^{BET} = k_{tip} v_i \tag{2.10}$$

dove

$$k_{tip} = \frac{1}{4}\left(\frac{1 + \sqrt{1 + \frac{64}{\sigma C_{l\alpha}}\theta_{75}/3}}{\theta_{75}/3}\right) \tag{2.11}$$

ottenuta incorporndo α_0 in θ_{75} per sezioni di profili con bassa curvatura della corda media. Ora è possibile stimare θ_{75}:

$$\theta_{75} = arctan\left(\frac{\Gamma}{0.75\pi D}\right) \tag{2.12}$$

Incorporando ora le equazioni (2.10) (2.11) (2.12) è possibile ottenere una funzione analitica per il coefficiente di trazione:

$$C_T = \sigma\pi \left(\frac{4\Gamma}{9\pi D} - \frac{1}{2k_{tip}} \right) \tag{2.13}$$

Il coeffiiente di merito dipende ora solo dai parametri $\frac{\Gamma}{D}$ e Re (σ è considerata costante) [13]

Dimostrato ciò è possibile sviluppare una funzione, che presenta solo le due variabili precedentemente citate, in grado di adattarsi al meglio ai dati teorici. Per fare ciò si ricorre all'uso di un metodo iterativo con il fine di minimizzare l'ordine del polinimio necessario. Tale funzione è considerata valida solo fino ad valore massimo del numero di Reynold, oltre al quale il flusso non è più approssimabile come ideale. Per estenderne il campo di utilizzo oltre al valore critico, è necessario implementare il contributo di fattori di correzione sperimentali. Questi sono derivati a partire da dati sperimentali ottenuti mediante test di laboratorio. I fattori di correzione operano una modifica nel valore delle costanti del polinomio ottenuto con l'ipotesi di fluido ideale. Ogni costante ,infatti, dipenderà a sua volta da fattori di correzione sperimentali.

$$f(Re) = f_0 + f_1 Re + f_2 Re^2 \tag{2.14}$$

$$\tag{2.15}$$

$$f_0 = (\frac{\Gamma}{D})^2 [f_{00} + f_{10}\frac{\Gamma}{D}) + f_{20}\frac{\Gamma}{D}^2] \tag{2.16}$$

$$f_1 = (\frac{\Gamma}{D})^2 [f_{01} + f_{11}\frac{\Gamma}{D}] \tag{2.17}$$

$$f_2 = (\frac{\Gamma}{D})^2 f_{02} \tag{2.18}$$

Comparando l'equazione ricavata mediante metodo iterattivo, che mette in luce la dipendenza da sole due variabili, con i dati sperimentali è emerso che questa caratterizza in modo opportuno le prestazioni dei multirotori presenti sul mercato. [14] [15]

Per l'analisi delle performance e la loro ottimazione si ha la necessità di descrivere e stimare il tempo di scarica della batteria, così da poter determinare il tempo di volo del multirotore.
In primo luogo si deve ricordare che nella seguente analisi verranno prese in considerazioni le tecnologie maggiormente utilizzate nei multirotori. Tra queste ricordiamo le batterie MH-Ni, Li-Po e a ioni di Litio. Queste, infatti, presentano una vita operativa lunga, un basso tasso di auto-scarica e un alto valore del rapporto energia/peso.
Considerando l'equazione trovata per la potenza erogata dalla batteria P_b e considerando che per il volo livellato tale valore è costante, si può proporre una nuova formulazione per il processo di scarica della batteria. In questo caso il tempo viene considerato come una funzione della capacità scaricata e della potenza erogata.
Prendendo in esame la corrente fornita dal pacco batteria I=I(t) al tempo t e la capacità elettrica C=C(t) si può ottenere che [16]

$$C(t) = \int_0^t I(s)\, ds \tag{2.19}$$

Dato che la potenza generata dalla batteria Pb avrà sempre valore positivo, il processo di scarica si interromperà al tempo t_f. Tale valore corrisponde all'istante in cui C_f=C(t_f) = K C_0 dove C_0

è la capacità nominale e K è una percentuale di scarica predefinita ¡1. Si può quindi notare che la batteria verrà considerata scarica quando avrà raggiunto un valore minimo prestabilito e non quando la capacità avrà valore zero.

Il tempo di scarica allora può essere espresso come:

$$t_{\mathrm{f}} = \delta P_{\mathrm{b}}{}^{\epsilon} C_{\mathrm{f}}{}^{\beta} \tag{2.20}$$

dove i coefficienti ϵ, δ e β dipendono dalla tecnologia della batteria presa in considerazione, dalla temperatura ambiente e dal numero di celle implementate in serie. Si ricorda che il valore dei coefficienti viene determinato sperimentalmente e che $\epsilon < -1, \delta > 0$ e $0 < \beta < 1$.
In fase di progetto preliminare per batterie Li-Po per cui non si ha a disposizione un'attrezzature per eseguire i test si può fare riferimento alle seguenti espressioni analitiche:

$$\delta_0(N_{\mathrm{s}}) = 0.1067 {N_{\mathrm{s}}}^3 + 0.8960 {N_{\mathrm{s}}}^2 + 2.488 N_{\mathrm{s}} + 0.6299 \tag{2.21}$$
$$\epsilon_0(N_{\mathrm{s}}) = 2.91710^{-4} {N_{\mathrm{s}}}^3 - 1.37510^{3} {N_{\mathrm{s}}}^2 + 3.08310^{3} N_{\mathrm{s}} - 1.041 \tag{2.22}$$
$$\beta_0(N_{\mathrm{s}}) == 0.9664 \tag{2.23}$$

dove N_{s} rappresenta il numero di celle collegate in serie e la temperatura di riferimento presa in esame corrisponde a 23°C. La variazione dei valori dei coefficienti che dipendono dal numero di celle implementate è legata anche alla tecnologia della batteria scelta e dall'invecchiamento della stessa. Per la scrittura delle formule appena enunciate si è scelto di considerare un pacco batteria di riferimento a metà della vita operativa.
Questo permette di ottenere una media tra le prestazioni ottimali, inizio vita, e quelle relative ad una condizione degradata.
Per ottenere una stima precisa del tempo di scarica si dovrebbero considerare le variazioni di ρ_0 e ϵ_0 al variare della vita della batteria monitorandone il comportamento a bordo.
Considerando però i dati sperimentali si è constatato che tali variazioni non influenzano in modo rilevante il tempo di volo quindi al fine dello studio svolto nel lavoro di tesi possono essere considerate trascurabili e si possono considerare valide le formule precedentemente descritte.

In secondo luogo, si deve considerare che la temperatura a cui la batteria opera ne altera le prestazioni. In particolare, si è studiato come primo caso quello relativo alla temperatura ambiente (τ_0=23°C) e successivamente si è presa in esame una temperatura inferiore (17°C) rilevando una regressione lineare delle prestazioni.
Si ha quindi la necessità di descrivere matematicamente tale variazione in funzione della differenza tra le temperatura operativa e quella ambiente $\Delta\tau$=τ- τ_0 e il numero di batterie collegate in serie.
Si ottiene quindi:

$$\delta(N_{\mathrm{s}}, \Delta\tau) = \delta_0(N_{\mathrm{s}})(1 - c_1\Delta\tau) \tag{2.24}$$
$$\epsilon(N_{\mathrm{s}}, \Delta\tau) = \epsilon_0(N_{\mathrm{s}})(1 - c_2\Delta\tau) \tag{2.25}$$
$$\beta(\Delta\tau) = \beta_0(1 - c_3\Delta\tau) \tag{2.26}$$

Dove i parametri c_1, c_2 e c_3 sono coefficienti di correzione ed equivalgono rispettivamente a: c_1=0.0046 1/∘C, c_2=0.0024 1/∘C, and c_3=0.0011 1/∘C.
Sommando le masse di tutti i componenti del multirotore e moltiplicando per l'accelerazione gra-

vitazionale costante g=9.80665 $\frac{m}{s}$ è possibile ottenere il peso totale al decollo. In questo modo, conoscendo la configurazione che il multirotore deve adottare durante il volo, è possibile ottenere la spinta necessaria. Utilizzando le formule relative al calcolo della potenza (2.5), è quindi possibile stimare la potenza elettrica richiesta per il volo e, di conseguenza, la potenza che la batteria deve essere in grado di erogare. Successivamente, utilizzando le formule per il calcolo del tempo di scarica (2.20), è possibile determinare il tempo di volo, permettendo così di valutare se le componenti considerate siano idonee a soddisfare le prestazioni richieste dalla missione.

2.3 Analisi su base sperimentale

Tale approccio differisce dal precedente in quanto non si baserà sulla formulazione di funzioni per il calcolo delle proprietà di interesse ma si baserà sull'uso di dati tabulati ricavati per via sperimentale. Questo è il metodo che verrà implementato nel lavoro di tesi. I dati sperimentali che verranno presi in esame si riferiscono alla componente propulsiva del multirotore, costituito da: motore elettrico, elica ed ESC. Per assicurare che tale metodo abbia un range di applicazione sufficiente si dovrà fare riferimento ad un numero appropriato di dati tabulati. A tale scopo si è preso in esame il database fornito dall'azienda Tyto Robotics [10], che descrive numerosi esperimenti condotti da diversi utenti e che si diversificano per tipologia di unità accoppiate tra loro.
È importante infatti avere a disposizione differenti tipologie di motore, eliche ed ESC e valutare come le performance variano a seconda di come esse vengono implementate tra loro. A tal propositi il database riporta in ordine la tipologia di motore, di elica e di ESC utilizzati per la conduzione dell'esperimento e fornisce anche informazioni aggiuntive sulle diverse tipologie. Si può infatti ricavare il peso totale del sistema propulsivo, dato dalla somma dei singoli pesi, il passo dell'elica, il suo diametro, la corrente massima supportata dall' ESC e altre specifiche di interesse.
Dopo aver eseguito l'esperimento vengono messi a disposizione sul sito differenti diagrammi che riportano i risultati ottenuti, ad esempio considerando un motore T-Motor AT2814 900KV, un'elica Graupner CAM Folding Prop 10X6 25X15 con diametro 10 in e passo 6 in ed un ESC Zubax Myxa A2 si otterranno i seguenti risultati:

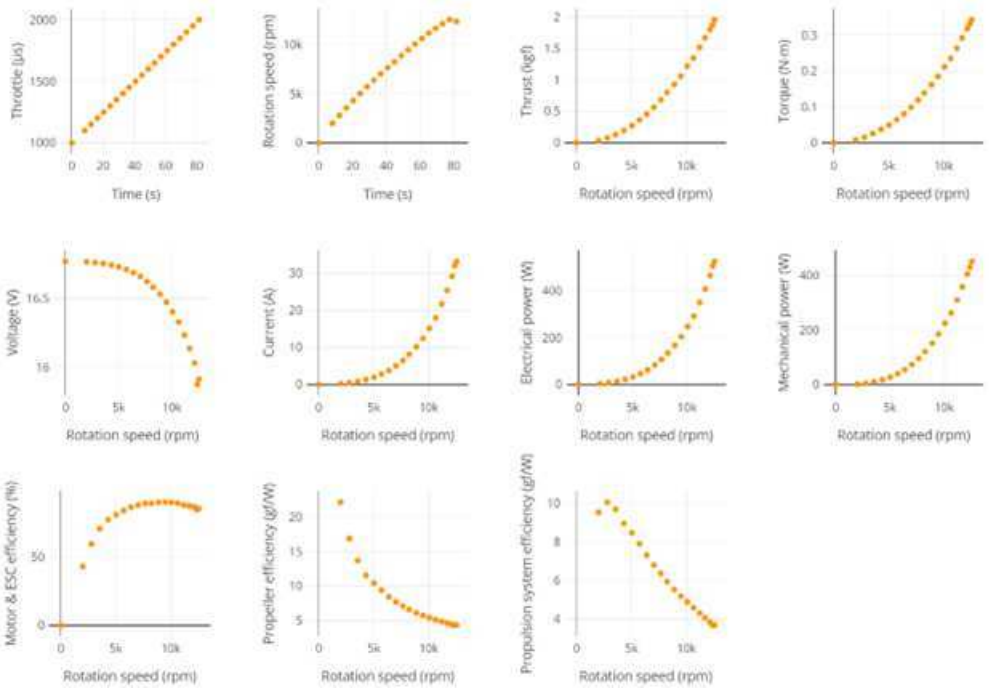


Figura 2.3: grafici disponibili nel database relativi a motore T-Motor AT2814 900KV, elica Graupner CAM Folding Prop 10X6 25X15 ed ESC Zubax Myxa A2

Gli stessi dati sono resi disponibili anche in froma tabulata:

Time (s)	Throttle (µs)	Rotation speed (rpm)	Thrust (kgf)	Torque (N·m)	Voltage (V)	Current (A)	Electrical power (W)	Mechanical power (W)	ESC efficiency (%)	Propeller efficiency (gf/W)	system efficiency (gf/W)
0	1000	0	0.0002	−0	16.77	0.0026	0.0432	0	0		
8.165	1100	1994	0.0355	0.0077	16.77	0.2223	3.728	1.605	43.06	22.1	9.515
12.25	1150	2791	0.0759	0.0154	16.77	0.4512	7.565	4.5	59.49	16.87	10.04
16.35	1200	3544	0.1301	0.0256	16.76	0.8023	13.45	9.504	70.68	13.69	9.675
20.42	1250	4300	0.1946	0.0374	16.75	1.299	21.76	16.83	77.34	11.57	8.944
24.51	1300	5029	0.2725	0.0496	16.73	1.927	32.25	26.13	81.02	10.43	8.452
28.6	1350	5719	0.3645	0.0648	16.72	2.764	46.2	38.82	84.01	9.39	7.889
32.65	1400	6380	0.4572	0.081	16.69	3.745	62.52	54.13	86.58	8.446	7.313
36.72	1450	7031	0.5643	0.0992	16.66	4.984	83.06	73.06	87.96	7.724	6.794
40.79	1500	7658	0.6816	0.1192	16.63	6.438	107.1	95.6	89.3	7.13	6.367
44.85	1550	8234	0.7996	0.1394	16.59	8.119	134.7	120.2	89.28	6.651	5.938
48.94	1600	8888	0.9274	0.1621	16.53	10.14	167.6	150.9	90.02	6.147	5.533
53.01	1650	9485	1.061	0.1855	16.47	12.42	204.6	184.3	90.05	5.759	5.186
57.07	1700	10063	1.218	0.2123	16.4	15.15	248.6	223.7	89.98	5.444	4.899
61.12	1750	10634	1.349	0.2354	16.33	17.95	293.1	262.1	89.43	5.144	4.601
65.18	1800	11185	1.517	0.2643	16.23	21.67	351.8	309.6	88	4.899	4.312
69.28	1850	11677	1.669	0.2927	16.14	25.36	409.2	357.9	87.46	4.664	4.079
73.38	1900	12139	1.803	0.3184	16.03	29.15	467.3	404.8	86.62	4.455	3.859

Figura 2.4: dati tabulati disponibili nel database relativi a motore T-Motor AT2814 900KV, elica Graupner CAM Folding Prop 10X6 25X15 ed ESC Zubax Myxa A2

I dati che verranno utilizzati in tale lavoro di tesi sono: velocità di rotazione, trazione, coppia torcente, voltaggio e potenza elettrica.

2.4 Processo operativo

Come visto in precedenza, conoscendo le masse degli elementi che compongono il sistema, è possibile ricavare la trazione che deve essere sviluppata per garantire l'equilibrio. Per ottenere tale valore però non si dovrà considerare solo il sistema propulsivo ma anche le altre parti che costituiscono il multirotore, ossia il payload, la massa della struttura e il contributo dato dalla necessità di implementare una batteria composta da un numero N_s di celle.
L'utente dovrà essere in grado di fornire tutti i dati elencati precedentemente e di conseguenza si otterrà il valore di trazione.

Conoscendo T, è possibile utilizzarla come input e usufruire delle tabelle disponibili per ricavare le restanti grandezze desiderate mediante un'interpolazione dei dati resi noti per via empirica.
Si deve però ricordare che i test condotti riportano un valore massimo della trazione oltre il quale non è più possibile utilizzare tale tecnologia. A tal proposito si è deciso di prendere in esame tre differenti casi considerando come riferimento un valore di sicurezza pari a metà del valore della trazione massima che il sistema è in grado di fornire:

- sistema adeguato: se la trazione necessaria al volo risulta minore od uguale al valore di sicurezza. Questa è considerata la condizione ideale per il volo
- sistema sottodimensionato: se la trazione necessaria al volo risulta maggiore del valore di sicurezza precedentemente stabilito ma inferiore al valore massimo che il sistema può fornire. In questo caso il volo risulta possibile ma sarebbe più opportuno scegliere un differente sistema propulsivo.
- errore: nel caso in cui il volo richieda che sia fornita una trazione superiore a quella massima che può esse erogata. In questo caso il volo risulta impossibile e si ha la necessità di adottare una tipologia di unità propulsiva differente.

Stabilita la tipologia di tecnologia implementata si è ora in grado di calcolare la potenza che la batteria deve erogare e, considerando la formula precedentemente descritta per il tempo di scarica, il tempo di volo.

Un ulteriore beneficio dell'implementazione di un codice in grado di leggere i dati tabulati e restituire come output i valori necessari per determinare le prestazioni di un multirotore consiste nella riduzione del tempo necessario ad eseguire i calcoli, come invece accade nell'analisi matematica. In secondo luogo, si noti come la possibilità di poter cambiare solo uno o più parametri rendano immediata la possibilità di studiare come un sistema sia in grado di rispondere a specifiche condizioni di volo, così da poter comprendere se esso sia adatto alla tipologia di missione che deve svolgere. In ultimo luogo si nota come la possibilità di cambiare la tipologia di sistema propulsivo, semplicemente cambiando la tabella da analizzare con un insieme di dati riferito ad un test differente, offra la possibilità di operare un confronto a priori riducendo il volume di calcolo e senza la necessità di acquistare le componenti prese in esame.

Capitolo 3

Implementazione Software

Tale lavoro di tesi verte sullo svulippo di un'applicazione che sia in grado di ricevere come input le caratteristiche strutturali del multirotore, le condizioni di volo e la tipologia di batteria implementata. Inoltre, deve offrire la possibilità di scegliere tra differenti unità propulsive testate in laboratorio. Ottenuti questi dati il programma dovrà restituire come output una stima della potenza necessaria per garantire il volo e una stima del tempo di volo.
Per la creazione di tale applicazione si è fatto riferimento ad App Designer ovvero ambiente di sviluppo integrato fornito da MATLAB.

3.1 App Designer

Il programma utilizzato per la scrittura del codice è App Designer, ovvero un ambiente di sviluppo integrato che consente la creazione di applicazioni che si interfacciano al programma MATLAB.
Una delle caratteristiche principali di tale sistema è la possibilità di ottenere un'interfaccia grafica utente (Graphical User Interface GUI). Con il termine GUI si identificano quegli strumenti di lavoro che permettono agli utenti di interagire con altri programmi, ad esempio MATLAB, utilizzando elementi grafici come finestre, pulsanti, caselle di testo, menù a discesa o altre componenti visive. Questo consente di poter eseguire operazioni più complesse senza avere la necessità di scrivere un codice per ogni sottounità, poichè esse presentano integrate le istruzioni che consentono il loro utilizzo. Uno dei maggiori benefici però risiede nella facilità di utilizzo delle GUI, in quanto per poter utilizzare l'applicazione basterà interfacciarsi con la parte grafica, e quindi sono di facile utilizzo anche per utenti che non possiedono conoscenze in programmazione. Questo rende il programma più accessibile e di più facile utilizzo.
App Designer in particolare è un'applicazione inclusa direttamente nell'IDE di MATLAB che semplifica la creazione di GUI interative per le proprie esigenze. Anch'essa consente agli utenti di trascinare e rilasciare gli elementi che risultano necessari per la creazione dell'applicazione nell'interfaccia utente. App Designer, infatti, mette a disposizioni differenti strumenti come ad esempio pulsanti, caselle di testo, grafici e molto altro che possono essere implementati, e sono quindi in grado di collaborare tra loro in modo simultaneo e coordinato, per eseguire le funzioni desiderate. Gli utenti sono in grado di definire il comportamento dell'applicazione associando funzioni MATLAB ad eventi, quali il clic di un pulsante o l'inserimento di differenti valori in una data casella. App Designer risulta uno strumento di facile utilizzo soprattutto per quanto concerne la possibilità di sviluppare un software senza la necessità di possedere conoscenze avanzate di programmazione. Questo è dato dalle sue caratteristiche principali ovvero:

- Editor di layout grafico: facilita la creazione dell'interfaccia che sarà poi visualizzata quando verrà utilizzata l'applicazione. L'utente infatti può posizionare manualmente e personalizzare gli elementi che compongono l'applicazione mediante un interfaccia drag-and-drop. Questa è una particolare tipologia di interfaccia che consente all'utente di interagire con gli strumenti trascinandoli da una posizione ad un'altra e rilasciandoli per eseguire un'operazione precisa.

- Editor di codice: è una ben definita porzione dell'interfaccia in cui è possibile scrivere un codice MATLAB per definire il comportamento dell'applicazione. Questo implica il dover stabilire quando e in che modo determinati elementi devono interagire tra loro. È inoltre possibile riservare una parte di codice per la gestione degli eventi, ovvero stabile quando determinate funzioni devono essere riprodotte e in che ordine le istruzioni dovranno essere eseguite.

- Visualizzazione in tempo reale: rende possibile visualizzare in tempo reale le eventuali modifiche apportate, durante la progettazione, in modo tale da poter ottenere un riscontro immediato e stabilire quale, tra le differenti opzioni disponibili, sia la più idonea al tipo di software che si vuole ottenere.

- Test e debug: sono resi disponibili appositi strumenti per testare ed eliminare eventuali errori direttamente nell'ambiente di sviluppo. Questo rappresenta un aiuto nella fase di programmazione del codice in quanto rende visibili mediante segnali di errore, con differenti colori in base all'entità del bug, le parti di codice che potrebbero essere incomplete o potrebbero risultare impossibili da riprodurre.

- Generazione automatica di codice: App Designer può generare automaticamente il codice MATLAB corrispondente alla GUI progettata, facilitando l'integrazione di un codice personalizzato o la condivisione del progetto con altri utenti. In questo modo si facilita anche l'estensione dell'applicazione per l'esecuzione di nuove funzioni.

Date queste sue caratteristiche App Designer risulta la tecnologia più efficiente per la tipologia di lavoro che si vuole svolgere in tale elaborato di tesi.

3.2 Descrizione dell'implementazione

Per l'implementazione del codice dell'applicazione si deve, in primo luogo, individuare i differenti input che devono essere forniti al sistema per consentire il calcolo delle prestazioni del multirotore.

Input

Per stabilire quali elementi siano necessari si è suddiviso il campo di inserimento degli input in quattro sottogruppi. Questi prendono in esame le principali sezioni di interesse per lo studio delle prestazioni:

- Parametri di massa: prendono in esame le diverse componenti che determineranno il peso complessivo del multirotore. Tra queste prenderanno parte le masse relative all'avionica, alla struttura e al carico utile. Si deve ricordare però che per il calcolo della trazione dovranno essere addizionati anche i pesi della batteria implementata e dell'unità propulsiva scelta.

- Parametri di volo: in questa sezione si avrà la possibilità di inserire la quota a cui si desidera effettuare il volo. Tale grandezza renderà possibile il calcolo della temperatura operativa e, in caso di esigenza, sarà possibile inserire un eventuale offset di temperatura.

- Gestione dell'alimentazione: tale zona di inserimento degli input serve a determinare la tipologia di batteria che si vuole implementare, per garantire la produzione di potenza necessaria al volo. In questa sezione sarà possibile inserire i valori numerici che fanno riferimento al numero di celle implementate, alla capacità nominale, alla massa totale della batteria, alle perdite di potenza date dall'assorbimento di energia da parte dell'avionica e quelle date dalla presenza di un eventuale carico utile.
- Unità propulsiva: l'ultimo gruppo si riferisce alla scelta che si deve fare nei confronti del sistema propulsivo da implementare. Si potrà infatti inserire il numero di rotori che si desiderano prendere in esame, gli angoli di tilt e dietro che dovranno essere mantenuti durante il volo ed, infine, si potrà scegliere la tipologia di motore elettrico, elica ed ESC selezionando uno dei vari file, contenenti i dati sperimentali tabulati, resi disponibli nell'applicazione.

È ora possibile studiare i vari componenti utili per l'implementazione e la programmazione dell'applicazione, studiandone le caratteristiche principali e il loro funzionamento. Per una comprensione più dettagliata di tali strumenti verranno riportati degli esempi pratici che fanno riferimento a tale lavoro di tesi.

Edit Field (Numeric)

Per permettere l'inserimento di valori numerici si deve ricercare nella libreria dei componenti lo strumento Edit Field (Numeric). Questo potrà essere selezionato e rilasciato nell'interfaccia così da poter essere richiamato nel processo di scrittura del codice. Esso è composto da due differenti componenti, l''Edit Field', ovvero una casella in cui è possibile inserire un testo per identificare la tipologia di variabile che si sta prendendo in esame, e la casella associata, che permette di poter inserire il valore che essa deve assumere durante l'utilizzo dell'app. Quando il campo di inserimento del valore viene modificato, App Designer è in grado di registrare tale cambiamento all'interno del codice stesso, così da poter apportare dei cambiamenti in merito al caso da esaminare senza dover variare il codice.

(a) edit field (numeric) nella component library

(b) edit field (numeric) nell'interfaccia

Figura 3.1: Edit Field (Numeric)

Per quanto concerne la massa del payload, ad esempio, si dovranno scrivere le seguenti righe di codice:

```
payload_mass=app.payloadmassEditField.Value;
```

in cui si noti come da una variabile 'payload_mass' viene associata la casella di inserimento che sarà visualizzata nell'interfaccia e, nella quale, sarà possibile inserire un dato valore.
In questo, modo ogni volta in cui si ha la necessità di modificare tale valore, sarà sufficiente operare nella casella dell'interfaccia e, in tempo reale, si avrà la modifica corrispondente all'interno del codice.

Figura 3.2: esempio pratico dell'utilizzo della componenete edit field (numeric)

Nella scrittura della porzione di codice, adibita ai calcoli per l'ottenimento dei risultati necessari, ci si dovrà riferire a tali grandezze mediante il nome della costante associata all' 'Edit Field (Numeric)' preso in esame.

Label

Per permettere l'inserimento di porzioni di testo si deve utilizzare questo specifico strumento. Lo scopo principale per cui è utilizzato è quello di aggiungere una descrizione ad un campo numerico. Come nel caso precedente, tale elemento deve essere individuato nella libreria dei componenti, selezionato e trascinato nell'interfaccia.

(a) label nella component library

(b) label nell'interfaccia

Figura 3.3: Label

In questo modo, cliccando sopra la casella, è possibile selezionarla e successivamente cambiarne il valore, senza dover modificare il codice del programma. Nella 'Component Browser' infatti è possibile selezionare la componente desiderata e, analizzando la sottosezione 'text', è possibile modificare il testo che si vuole far apparire.

Figura 3.4: sezione text relativa alla casella label

Considerando l'esempio precedente, è ora possibile aggiungere, mediante l'uso di tale componente, una casella di testo per specificare il tipo di unità di misura utilizzata. In questo modo si rendere noto il sistema di misura adottato dall'applicazione e che, quindi, deve essere rispettato per l'ottenimento di risultati corretti.

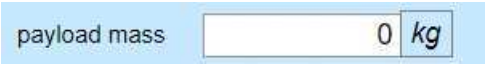


Figura 3.5: esempio pratico dell'utilizzo della componenete label

Drop Down

Tale componente consente di implementare nell'interfaccia dell'applicazione un menù a discesa. Questo permette all'utente che utilizza l'applicazione di selezionare una singola opzione da un elenco di proposte disponibili. Tali opzioni sono rese visibili solo nel momento in cui viene selezionata la casella di interesse, così da poter permetterne la scelta prima di richiudersi mostrando solo l'opzione selezionata.
Anche in questo caso, per aggiungere tale componente la si dovrà ricercare nella libreria, selezionarla, e trascinarla fino alla zona dell'interfaccia dove si desidera sia visualizzata.

(a) drop down nella component library

(b) drop down nell'interfaccia

Figura 3.6: Drop Down

Questa componente, nell'ambito del lavoro di tesi, è stata utilizzata per permettere la scelta dell'unità propulsiva da utilizzare. Si sono infatti rese disponibili differenti tipologie di unità, che differiscono tra loro a seconda della tipologia di motore elettrico, elica ed ESC implementati, e le cui prestazioni sono state rese disponibili per via sperimentale.
Per ottenere il risultato desiderato si è avuta, in primo luogo, la necessità di creare un file che fosse compatibile con MATLAB, in modo tale da poterne assicurare la corretta lettura durante l'utilizzo dell'applicazione. A tal proposito si è scelto di fare riferimento a file con estensione '.mat'. Questi file sono comunemente utilizzati per memorizzare variabili MATLAB, strutture di dati e altre informazioni in un formato binario compatto.
È stato quindi necessario copiare i dati forniti dagli utenti che hanno condotto i test in laboratorio, e che hanno successivamente reso disponibili nel database di Tyto Robotics, per renderli nel formato ricercato.
Questo è stato possibile mediante la creazione di uno script MATLAB in grado di creare una tabella contenente i dati, precedentemente ricavati dall'archivio digitale. Si può notare che questo insieme di dati è composto da cinque differenti colonne che indicano, in ordine: velocità di rotazione, trazione generata, coppia generata, voltaggio e potenza elettrica assorbita. Queste corrispondono alle grandezze che saranno successivamente necessarie per il calcolo degli output desiderati.

untitled4.m

```
tabella=[0.000000000    0.000137384 -0.00014175936154627    16.739807663    0.257163370754134;
1982.000000000  0.024245208 0.0054655283043033  16.733249283    2.724600006;
2823.000000000  0.05324941849631832 0.011475903247528   16.726148834    5.701812385;
3635.000000000  0.09186832723714011 0.018860453756999   16.715682144    9.983762083 ;
4419.000000000  0.14953110896512062 0.027390485375387   16.701502380    16.359980536;
5149.000000000  0.2175857588401136  0.037938789643407   16.681882553    24.604058664;
5865.000000000  0.28618659666290736 0.048594209109434   16.657930222    34.658147182;
6560.000000000  0.3659954481052347  0.060009351717803   16.626174698    46.571416317;
7222.000000000  0.46016368500098404 0.073316729837064   16.587986298    61.030195689;
7883.000000000  0.5445738905715408  0.08630811542245    16.545069885    77.025178259;
8533.000000000  0.6386977195032351  0.09989365971785    16.492250748    95.328634931;
9151.000000000  0.7426031489948046  0.11607535169172    16.430558319    117.632561058;
9742.000000000  0.8694498957264306  0.13365096330914    16.360534363    142.698933088;
10327.000000000 0.9810899292585133  0.15217385899728    16.282948837    171.183308733;
10926.000000000 1.088569788 0.16808461863809    16.192393188    200.787870192;
11536.000000000 1.237145849 0.1877749904335 16.089912262    237.560875992;
12121.000000000 1.353245780 0.20691493813443    15.983175850    273.267153147;
12664.000000000 1.496301219 0.22518243522393    15.866478577    312.430979061;
13194.000000000 1.619870811 0.24532078186176    15.743029594    355.093444305;
13214.000000000 1.622332707 0.24466635449646    15.703138962    357.115989978];
```

Figura 3.7: esempio pratico script MATLAB

Dopo aver riprodotto lo script, è possibile salvare tale tabella con un'stensione '.mat' semplicemente utilizzando la funzione 'save' di MATLAB.
Essendo ora in possesso di un file idoneo alla lettura è possibile utilizzare la funzione 'load', nell'apposita sezione di App Designer dedicata alla scrittura del codice, per permettere al programma di utilizzare tali valori durante la fase di calcolo.
Per permettere alla casella Drop Down di riconoscere le diverse unità propulsive disponibili, si deve in primo luogo denominarle nella 'Component Browser' alla sezione 'items'. Questa permette

di aggiungere, eliminare o cambiare l'ordine di comparsa delle differenti opzioni che si vogliono rendere disponili.

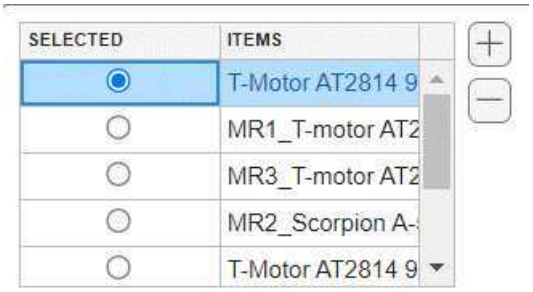


Figura 3.8: menù reltivo alle opzioni da implementare nella componente drop down

Si deve però ricordare che per associare dei valori, o delle funzioni, a ciascuna delle opzioni che si vogliono rendere disponili nel software, si ha la necessità di operare nel codice dell'applicazione. A tal proposito, facendo riferimento al lavoro di tesi, si nota che per ogni unità propulsiva deve corrispondere un differente file di dati da analizzare e una differente massa, data dalla somma delle tre componenti prese in esame durante la conduzione dell'esperimento.

Per la scrittura del codice si è fatto riferimento a due distinti costrutti di controllo di tipo 'switch'. Per associare ad ogni unità la rispettiva massa totale sono state implementate le seguenti righe di codice:

```
switch app.propulsionunitDropDown.Value


    case 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop 10X5 25X12_Zubax Myxa A2'
         app.propulsionunitmassEditField.Value=0.162;
    case 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop 10X6-3 25X16_ Zubax Myxa A2'
         app.propulsionunitmassEditField.Value=0.162;
```

dove inizialmente si identifica in quale casella Drop Down si vuole inserire l'informazione. Successivamente si deve specificare, attraverso la dicitura 'case', il nome dell'opzione alla quale si vuole associare l'informazione e, in ultimo luogo, si associa alla massa il valore numerico.

Per associare ad ogni unità propulsiva il relativo file con stensione '.mat' contenente la tabella dati:

```
switch app.propulsionunitDropDown.Value


    case 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop 10X5 25X12_Zubax Myxa A2'
       load T-MotorAT2814900KV_GraupnerCAMFoldingProp10X525X12_ZubaxMyxaA2.mat;

    case 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop 10X6-3 25X16_ Zubax Myxa A2'
        load T-MotorAT2814900KV_GraupnerCAMFoldingProp10X6-325X16_ZubaxMyxaA2.mat;
```

che differisce dal caso precedente per la presenza dell'utilizzo della funzione 'load', precedentemente descritta.
In questo modo si è reso possibile l'inserimento di tutti i valori di input necessari per il calcolo delle prestazioni mediante l'interfaccia grafica.
L'interfaccia con cui l'utente di dovrà interagire risulta quindi essere:

Processo di calcolo

Per ottenere i risultati desiderati, si ha la necessità di programmare opportune righe di codice per determinare come le informazioni, fornite dagli input, debbano essere utilizzate ed elaborate tra loro.
Anche in questo caso si è deciso di suddividere il lavoro in base a macro-argomenti per rendere il codice di più facile lettura e comprensione.
In primo luogo, si ha la necessità di calcolare la massa totale del multirotore. Per ottenere ciò si devono richiamare i valori inseriti nelle caselle, precedentemente implementate nell'interfaccia, e relative alle varie masse dei sottosistemi presi in esame. Ad ognuna di queste verrà associato il nome di una variabile che potrà, allora, essere utilizzata nella scrittura del codice relativo ai calcoli da effettuare. Si deve inoltre ricordare che la massa del sistema propulsivo fa riferimento ad un'unica unità e, per tale motivo, si ha la necessità di moltiplicare tale valore per il numero di rotori implementati.

$$M_{\text{sisitema propulsivo}} = N M_{\text{singola unità}} \tag{3.1}$$

$$M_{\text{totale}} = M_{\text{payload}} + M_{\text{struttura}} + M_{\text{avionica}} + M_{\text{sisitema propulsivo}} + M_{\text{batteria}} \tag{3.2}$$

Essendo in possesso di tale valore si è in grado di calcolare il peso complessivo del velivolo moltiplicando la massa totale per la costante di gravità

$$\tag{3.3}$$

$$W = M_{\text{totale}} \cdot 9.80665 \tag{3.4}$$

Ora è possibile calcolare la trazione che ogni rotore deve fornire per garantire il volo, ricordando la formula:

$$T = \frac{W}{N \cdot \cos(\Gamma) \cdot \cos(\xi)} \tag{3.5}$$

Conoscendo la trazione, si utilizza tale dato per estrapolare dalla tabella, contenente i dati relativi agli esperimenti condotti sulle unità propulsive, i rispettivi valori di output ricercati.
Come anticipato precedentemente, si è deciso di fare riferimento a tre casi distinti operando un confronto tra il valore di T trovato, quello massimo erogabile dal sistema propulsivo ed un valore di sicurezza pari a Tmax/2. Per implementare tale distinzione nel codice del programma, si è fatto riferimento ad un ciclo 'if' che restituisce un differente segnale a seconda della condizione trovata:

```
        a=max(thrust);
        asicurezza=a/2;

    if   T_rot<=asicurezza
        app.warning.Value='adeguato';

    elseif  T_rot<=a
        app.warning.Value='sottodimensionato';

    else
        app.warning.Value='errore';


    end
```

Nell'ultimo caso non si ha la necessità di proseguire con i calcoli in quanto, non essendo la trazione in valore sufficiente, si può direttamente determinare che il volo non sarà possibile.
Di conseguenza, non si ha la necessità di proseguire con lo studio di tale casistica.
Nei restanti due casi, invece, il volo risulta ammissibile e le modalità per determinare i valori di output ricercati sono le stesse. Per questo motivo, verrà riportato solo lo studio del caso più ottimale ovvero quello relativo ad un valore di trazione adeguato.
In primo luogo, si devono richiamare i valori di trazione resi disponibili dai dati sperimentali, ovvero estrapolare dalla tabella dati la seconda riga di valori. Questo è possibile creando una variabile, nel nostro caso thrust1, e associandone solamente i dati relativi alla seconda colonna della tabella importata precedentemente. Si deve però notare che l'unità di misura relativa ai dati sperimentali è kgf e non N quindi, per esprimere tale dato nel sistema internazionale, si ha la necessità di moltiplicarlo per la costante di gravità considerata uguale a 9.80665 $\frac{m}{s^2}$.
Seguendo lo stesso principio spiegato precedentemente, si importano anche i dati relativi a potenza elettrica, voltaggio, velocità di rotazione e coppia.

Si può quindi affermare che ora l'applicazione sia in possesso di tutte le informazioni necessarie. Per il valore di trazione trovato in precedenza, ovvero quello che garantisce il volo, il software è in grado di restituire i valori di potenza elettrica, voltaggio, velocità di rotazione e coppia associati. Questo avviene mediante l'interpolazione dei singoli valori sperimentali per mezzo di una funzione che verrà poi confrontata con il valore di trazione.
Per fare ciò si utilizza lo strumento 'interp1' di MATLAB ovvero una funzione di interpolazione unidimensionale che viene utilizzata per interpolare i valori di una variabile dipendente in base a una variabile indipendente. Il primo valore determina quale sia la variabile indipendente, il secondo la variabile dipendente e l'ultimo specifica il punto in cui si desidera stimare il valore interpolato. Questa funzione può essere utile quando si desidera stimare valori intermedi tra i punti dati noti, come nel caso preso in esame nel lavoro di tesi.

```
thrust1=tabella(:,2);
thrust=thrust1*9.80665;
a=max(thrust);
asicurezza=a/2;

if   T_rot<=asicurezza

     pe=tabella(:,5);
     Vv=tabella(:,4);
     rotspeed=tabella(:,1);
     tor=tabella(:,3);

     Pel=interp1(thrust1,pe,T_rot/9.80665);
     V=interp1(thrust1,Vv,T_rot/9.80665);
     rot=interp1(thrust1,rotspeed,T_rot/9.80665);
     trq=interp1(thrust1,tor,T_rot/9.80665);
```

Anche in questo caso si deve prestare attenzione al tipo di unità di misura utilizzate. I dati sperimentali, infatti, sono espressi in kgf e ne consegue che, per effettuare un confronto, anche il valore di trazione necessaria al volo dovrà essere espressa con questa unità di misura e quindi, si ha la necessità di dividere tale valore per la costante di gravità.

Come ultimo output ricercato figura il tempo di volo, che è rappresentato dal tempo di scarica della batteria. In primo luogo, si deve ricordare che esso dipende dalla temperatura, il cui valore deve quindi essere stimato durante la fase operativa.
Per fare ciò si osserva che la temperatura varia con la quota di volo e, analizzandone l'andamento, è possibile stabilirne il valore conoscendo solo l'altezza a cui si desidera far operare il velivolo.
Essendo tale lavoro di tesi incentrato sull'utilizzo di multirotori, si è deciso di esaminare solo le quote che competono a tale tecnologia ovvero la troposfera (0 km -11 km) e la tropopausa (11 km – 20 km).

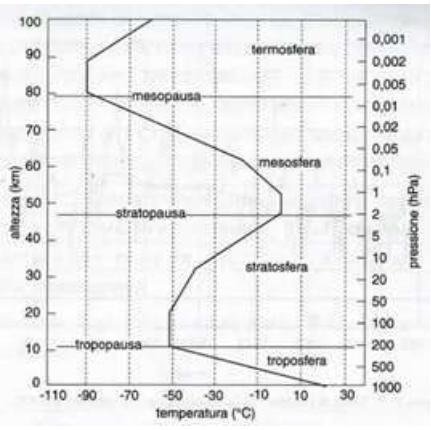


Figura 3.9: andamento della temperatura al variare della quota

Per la troposfera è possibile osservare che la dipendenza della temperatura con l'altezza segue la funzione $\Delta T = \lambda \cdot \Delta h$ dove λ assume un valore pari a 0.0065. Se ne deduce quindi che la temperatura tenderà a diminuire a partire da un valore di 15°C, a livello del mare, fino a -56.5°C corrispondente a 11 km di quota.
La tropopausa invece è considerabile come uno strato isotermo ovvero la temperatura resta costante all'aumentare dell'altezza. Come valore di riferimento per tale zona viene preso quello relativo a

11 km in troposfera, ovvero -56.5°C.
Tali affermazioni però si basano sull'assunzione che la temperatura a livello del mare sia pari a 15°C. Questo però risulta essere un valore approssimato in quanto può variare a seconda di differenti condizioni ambientali e quindi, si ha la necessità di implementare un offset nel calcolo della temperatura. L'offset rappresenta un valore che si va ad addizionare al risultato trovato e che tiene conto di tutte le possibili variabili, così che il calcolo effettuato rappresenti meglio il valore reale della temperatura operativa.
L'implementazione avviene attraverso l'uso di un ciclo 'if' descritto nelle seguenti righe di codice:

```
h=app.flightaltitudeEditField.Value;
dt=app.temperatureoffsetEditField.Value;

if h<11000
    temp1=288.15-0.0065*h;
    temp=(temp1+dt)-273.15;
else
    temp=(216.65+dt)-273.15;
end
```

Si deve prestare attenzione al fatto che, per operare tali calcoli, si devono creare due differenti variabili, quota operativa e offset, che richiamino le caselle corrispondenti dell'interfaccia grafica. In questo modo si possono far variare i valori di quota ed offset operando direttamente nell'interfaccia. Per ottenere il valore del tempo di volo, ovvero il tempo i scarica della batteria, è sufficiente implementare le formule descritte nel capitolo precedente.
Inizialmente si devono calcolare i parametri δ, ϵ, β:

$$\delta_0(N_\mathrm{s}) = 0.1067{N_\mathrm{s}}^3 + 0.8960{N_\mathrm{s}}^2 + 2.488N_\mathrm{s} + 0.6299 \tag{3.6}$$
$$\epsilon_0(N_\mathrm{s}) = 2.91710^{-4}{N_\mathrm{s}}^3 - 1.37510^3{N_\mathrm{s}}^2 + 3.08310^3N_\mathrm{s} - 1.041 \tag{3.7}$$
$$\beta_0(N_\mathrm{s}) = 0.9664 \tag{3.8}$$

dove

$$\delta_0(N_\mathrm{s}) = -0.1067{N_\mathrm{s}}^3 + 0.8960{N_\mathrm{s}}^2 + 2.488N_\mathrm{s} + 0.6299 \tag{3.9}$$
$$\epsilon_0(N_\mathrm{s}) = 2.91710^{-4}{N_\mathrm{s}}^3 - 1.37510^{-3}{N_\mathrm{s}}^2 + 3.08310^{-3}N_\mathrm{s} - 1.041 \tag{3.10}$$
$$\beta_0(N_\mathrm{s}) = 0.9664 \tag{3.11}$$

E successivamente determinare il tempo di volo secondo la formula:

$$t_\mathrm{f} = \delta {P_\mathrm{b}}^{\epsilon} {C_\mathrm{f}}^{\beta} \tag{3.12}$$

dove la potenza della batteria è ricavata sommando la potenza elettrica totale, la potenza dissipata dall'avionica e quella dissipata dal payload. Bisogna però prestare attenzione al fatto che il valore di potenza elettrica, ricavato dall'interpolazione dei dati numerici, fa riferimento solamente ad un'unità propulsiva e quindi, per ottenere il valore totale, si deve moltiplicare il singolo per il numero di rotori implementati N. Si ottiene:

$$P_\mathrm{batteria} = (N \cdot P_\mathrm{elettrica}) + P_\mathrm{avionica} + P_\mathrm{payload} \tag{3.13}$$

Il codice è ora da considerarsi completo in quanto sono state implementate tutte le informazioni

necessarie per il calcolo degli output richiesti.

Output

L'ultima operazione da svolgere per l'ottenimento di un'applicazione funzionante è quella relativa alla visualizzazione degli output nell'interfaccia grafica.
In primo luogo, si nota che si ha la necessità di implementare uno specifico componente che, quando viene selezionato, riproduce il codice. Per adempiere a tale funzione si è utilizzato un componente Button.

Button

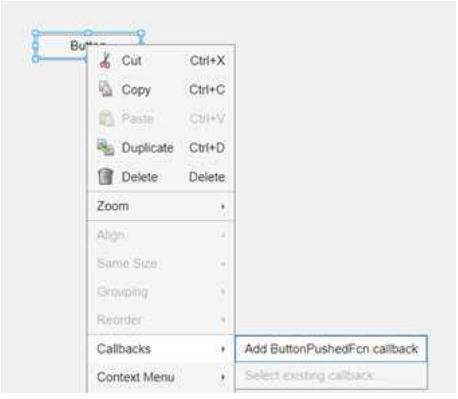
Per implementare questo elemento nell'interfaccia grafica basta selezionarlo dalla libreria di componenti e trascinarlo nella zona dell'interfaccia dove si desidera sia visualizzato.

(a) button nella component library

(b) button nell'interfaccia

Figura 3.10: Button

Per associare l'azionamento del bottone alle funzioni che vogliono far eseguire si deve, in primo luogo, creare una callback, ovvero creare un'apposita sezione del codice associata a tale componente. Selezionando, infatti, l'opzione 'Add Button PushedFcn callback' si creerà automaticamente una funzione nella sezione relativa alla scrittura di codice. Ora le informazioni e azioni inserite all'interno di tale funzione verranno eseguite automaticamente quando si cliccherà sopra il bottone associato nell'interfaccia grafica.



(a) callback nella component library

```
% Button pushed function: Button
function ButtonPushed(app, event)

end
```

(b) scrittura del codice relativo alla componenete button

Figura 3.11: programmazione della componente Button

Se ne deduce quindi che tutte le componenti e le righe di codice, descritte precedentemente, dovranno essere riportate all'interno della sezione della funzione per permettere un corretto utilizzo dell'applicazione.

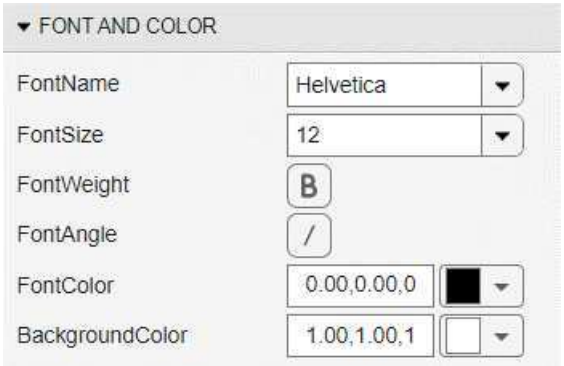
Ora l'applicazione è in grado di acquisire dati dall'esterno e utilizzarli, elaborandoli tra loro, per eseguire i calcoli richiesti. Si ha ora la necessità di rendere tali output visibili nell'interfaccia, così da renderli disponibili all'utente che desidera utilizzare il software.

Si ricorda che, nelle fase di scrittura del codice, per ogni output richiesto sono state associate delle variabili che assumono il valore numerico della grandezza che si desidera conoscere. Per trasportare tale grandezza nell'interfaccia si deve utilizzare la componente Edit Field (Numeric) precedentemente illustrata.
In termini operativi, si nota che si deve in primo luogo creare una casella che sarà visualizzata durante l'utilizzo dell'applicazione e, successivamente, si deve associare a tale casella il valore della variabile che si desidera far comparire:

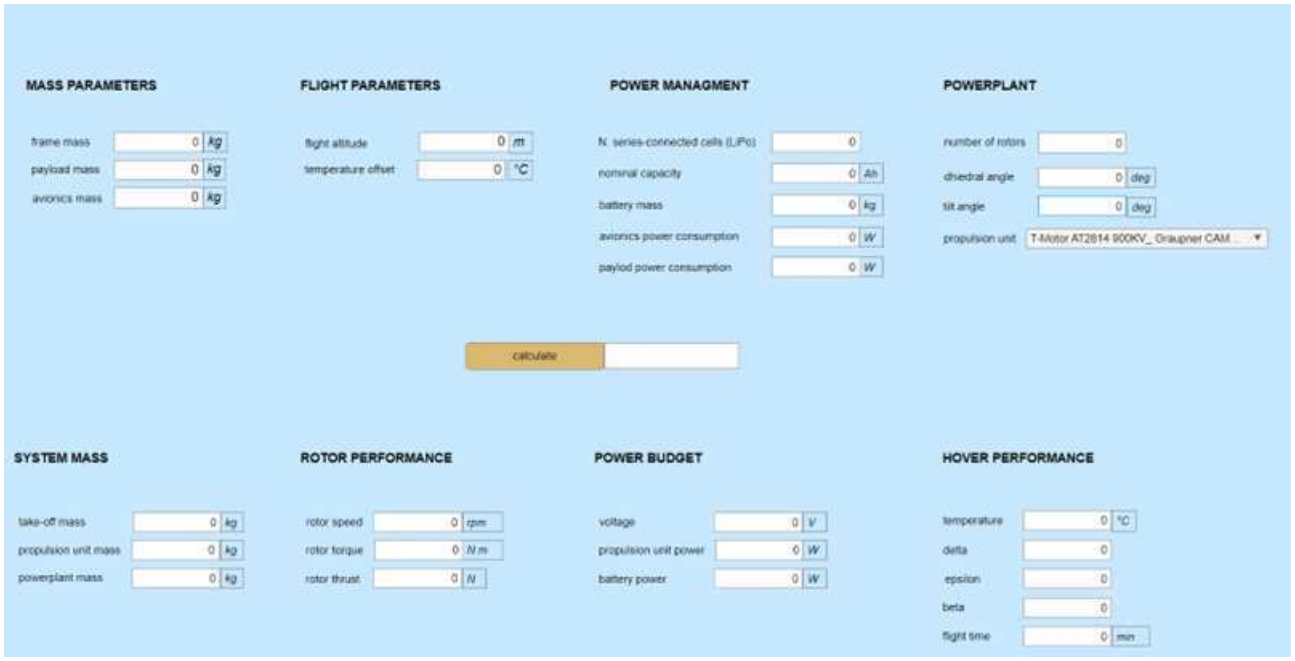
```
app.propulsionunitpowerEditField.Value=Pel;
app.voltageEditField.Value=V;
app.rotorspeedEditField.Value=rot;
app.rotortorqueEditField.Value=trq;
```

L'unica componente che deve essere implementata in maniera differente è quella relativa al segnale di attenzione che stabilisce, confrontando i valori di trazione, se il multirotore sia adeguato, sottodimensionato oppure se il volo non sia possibile.
Per tale componente si deve prendere in considerazione un Edit Field (Text). Nell'implementazione si comporta come la componenete Edit Field (Numeric) con l'unica differenza che fa riferimento a valori letterali e non numerici.
In questo modo si conclude lo sviluppo dell'applicazione che prevede sia una parte di programmazione, per determinare il tipo di operazioni da eseguire, l'ordine con cui devono essere svolte e il tipo di informazioni che si vogliono ottenere, sia una parte di progettazione dell'interfaccia, che renda intuitiva e di facile utilizzo l'applicazione.

Per quanto concerne il design dell'interfaccia si può facilmente modificare la posizione delle componenti trascinandole nella zona in cui si desidera siano visualizzate. Per operare in modo più specifico sul testo ed effettuare una scelta cromatica dei componenti si deve selezionare l'elemento da modificare nel Component Browser e, successivamente, fare riferimento alla sezione 'font and color' associata.



L'interfaccia grafica relaativa al software implementato risulta essere:



Capitolo 4

Esempi numerici

Nel capitolo precedente sono stati illustrati i differenti passaggi necessari per lo sviluppo dell'applicazione e i diversi componenti utilizzati per ottenere un programma in grado di ricavare le prestazioni di un dato multirotore, a partire da specifici input resi disponibili dall'utilizzatore.

In questo capitolo si cercherà di dimostrare che i dati ottenuti dall'utilizzo dell'applicazione possiedono una precisione sufficiente da rappresentare in modo ottimale la realtà. Questo renderà l'uso di tale applicazione idoneo alla progettazione preliminare di un multirotore. Si ricorda, infatti, che i parametri ricercati sono la potenza elettrica totale, ovvero quella che deve essere fornita dalla batteria, e conseguentemente il tempo di volo, dato dal tempo di scarica della batteria stessa. L'applicazione rende disponibili anche altri parmetri che possono risultare utili in fase di progettazione come, ad esempio, la temperatura operativa, la velocità di rotazione dell'albero motore e la coppia generata.

Per poter stabilire se i risultati ottenuti attraverso l'utilizzo del programma implementato siano idonei, si è scelto di prendere in esame tre differenti esempi di multirotore che differiscono per dimensione. Queste tecnologie sono state precedentemente studiate, attraverso un'analisi aeromeccanica, per conoscere il valore di potenza della batteria date specifiche condizioni di volo. In questo modo, sarà possibile confrontare il valore trovato per via analitica con quello ritrovato per via sperimentale.

Nell'esecuzione di tale studio però si sono riscontrate delle limitazioni in quanto l'analisi teorica prendeva in considerazione dei multirotori con unità propulsive che non risultano disponibili nel database di Tyto Robotics. È stato quindi effettuato un confronto tra tutti i dati disponibili e sono state selezionate le unità propulsive possedenti le caratteristiche geometriche e prestazionali che più si avvicinavano al caso teorico preso in considerazione.
Se ne deduce, quindi, che nel calcolo della potenza elettrica necessaria si attenderà un valore che approssima quello teorico ma che differisce lievemente da esso, poiché si stanno considerando due tecnologie simili ma non uguali.

Vengono riportati ora i tre differenti casi, enunciandone prima lo studio mediante il metodo aerodinamico e, successivamente, confrontandolo con i dati elaborati dall'applicazione.

4.1 Multirotore di piccole dimensioni (DJI PHANTOM 4)

Come primo esempio, prendiamo in considerazione un rotore di piccole dimensioni che possiede caratteristiche simili al DJI Phantom 4 V2.0.
Possiamo considerare un numero di rotori pari a N=4 dotati di eliche APC a due pale 9x4,5E ovvero con diametro pari a 9in e passo pari a 4,5in ($corda_{media}$=0.019 m e $corda_{75}$=0.022 m).
Per quanto concerne le condizioni di volo, viene considerata una quota pari a 10 m con una configurazione che prevede un angolo di diedro di 8 gradi ed un angolo di tilt pari a 3 gradi. Si considera un offset di temperatura pari a 0°C.
Per quanto riguarda l'alimentazione viene implementata una batteria Li-Po con tensione nominale pari a V_0=14,8 V per un numero di celle implementate pari a N_s=4 e capacità C_0=5,9 Ah.
L'unità propulsiva è ottenuta implementando un ESC da 12A e motori brushless Flyduino X2208.
La massa totale al decollo risulta quindi pari a 1,375 kg. Si considera inoltre una dissipazione di potenza da parte dell'avionica pari a 5W.

Considerando $\frac{\Gamma}{D}$= 0,5 e σ=0.106, dall'analisi aeromeccanica si è in grado di determinare che la coppia fornita dal motore è pari a 0,05Nm con una velocità angolare pari a 5600 rpm. La potenza che deve essere fornita dalla batteria risulta quindi pari a 161W. Il coefficiente di merito risulta f=0.644 [17]

Considerando ora di voler risolvere tale problema mediante l'utilizzo dell'applicazione si ha, in primo luogo, la necessità di determinare il sistema propulsivo più adatto a rappresentare il drone studiato. Per questo scopo viene scelto l'insieme di dati disponibili nel database di Tyto Robotics che implementa i seguenti sistemi:

- motore T-motor AT2814900KV: viene specificato che tale motore presenta un diametro dell'albero pari a 5mm, 12 poli magnetici ed un peso di 107g.
- elica Graupner Cam-Carbon Z 10X5 25X12: viene specificato che tale elica possiede un diametro di 10 in ed un passo di 5 in.
- ESC Zubax Myxa A2 : viene specificato che tale componente possiede un peso di 55g.

In seguito, vengono riportati i dati relativi ai test eseguiti e resi disponibili per l'unità propulsiva precedentemente illustrata, facendo riferimento sia a quelli espressi in forma grafica che alla loro versione tabulata.

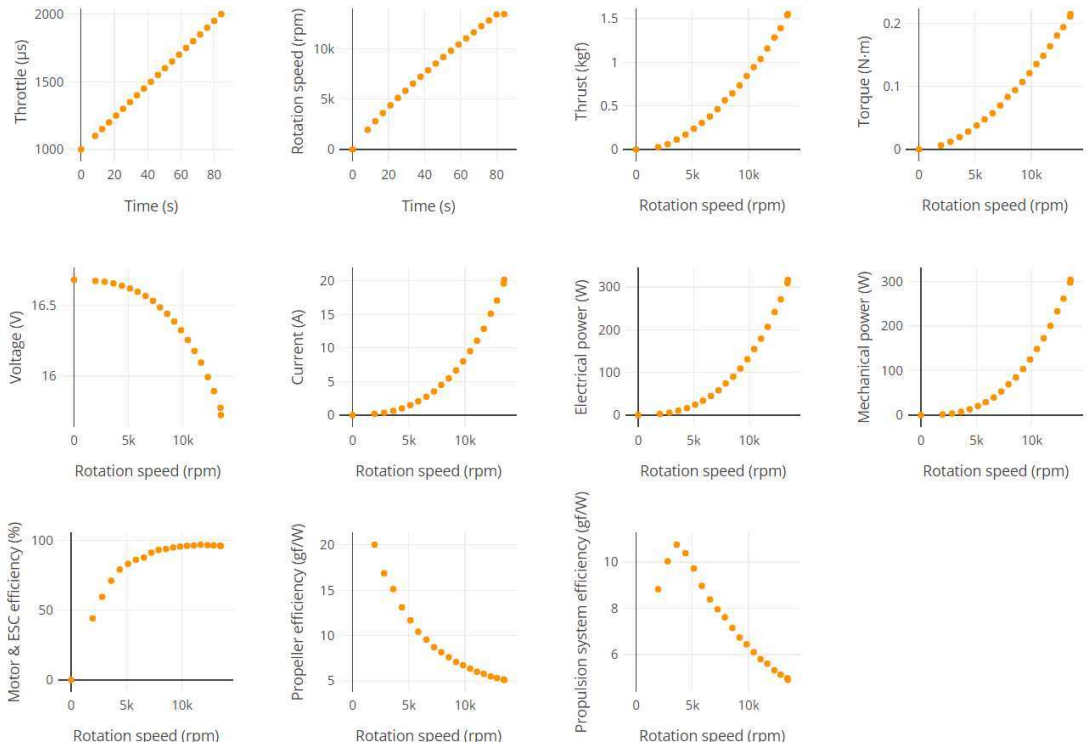


Figura 4.1: grafici dei test relativi al sistema propulsivo di un velivolo di dimensioni ridotte

Time (s)	Throttle (µs)	Rotation speed (rpm)	Thrust (kgf)	Torque (N·m)	Voltage (V)	Current (A)	Electrical power (W)	Mechanical power (W)	ESC efficiency (%)	Propeller efficiency (gf/W)	system efficiency (gf/W)
0	1000	0	0.0005	~0	16.68	-0.0062	0.1036	0	0		
8.392	1100	1955	0.026	0.0063	16.67	0.1764	2.942	1.296	44.06	20.02	8.822
12.59	1150	2804	0.0601	0.0121	16.67	0.3594	5.99	3.562	59.46	16.87	10.03
16.8	1200	3616	0.1127	0.0197	16.66	0.6293	10.48	7.444	71.02	15.13	10.75
20.99	1250	4388	0.1711	0.0284	16.64	0.9905	16.48	13.03	79.07	13.13	10.38
25.17	1300	5136	0.2386	0.0379	16.62	1.477	24.54	20.39	83.09	11.7	9.72
29.35	1350	5841	0.3041	0.0477	16.6	2.043	33.91	29.16	85.97	10.43	8.968
33.54	1400	6564	0.3776	0.0574	16.57	2.719	45.05	39.48	87.64	9.564	8.381
37.76	1450	7242	0.4621	0.0698	16.53	3.513	58.08	52.92	91.12	8.731	7.956
41.98	1500	7890	0.5642	0.0835	16.49	4.498	74.15	68.95	92.99	8.182	7.609
46.18	1550	8559	0.6441	0.0942	16.44	5.481	90.11	84.42	93.69	7.63	7.149
50.42	1600	9206	0.734	0.1072	16.39	6.648	108.9	103.3	94.84	7.105	6.738
54.66	1650	9833	0.8418	0.1211	16.32	8.003	130.6	124.7	95.44	6.752	6.444
58.89	1700	10457	0.945	0.1357	16.26	9.52	154.7	148.6	96.01	6.361	6.107
63.12	1750	11066	1.038	0.1487	16.18	11.07	179.1	172.3	96.22	6.027	5.799
67.33	1800	11671	1.159	0.1638	16.1	12.85	206.8	200.2	96.82	5.789	5.605
71.55	1850	12292	1.284	0.1809	15.99	15.09	241.3	232.9	96.52	5.512	5.32
75.78	1900	12850	1.392	0.1941	15.89	17.06	271.2	261.2	96.32	5.328	5.132

Figura 4.2: dati tabulati relativi al sistema propulsivo di un velivolo di dimensioni ridotte

È ora possibile compilare i campi necessari nell'applicazione seguendo le specifiche precedentemente illustrate e ottenendo i seguenti risultati:

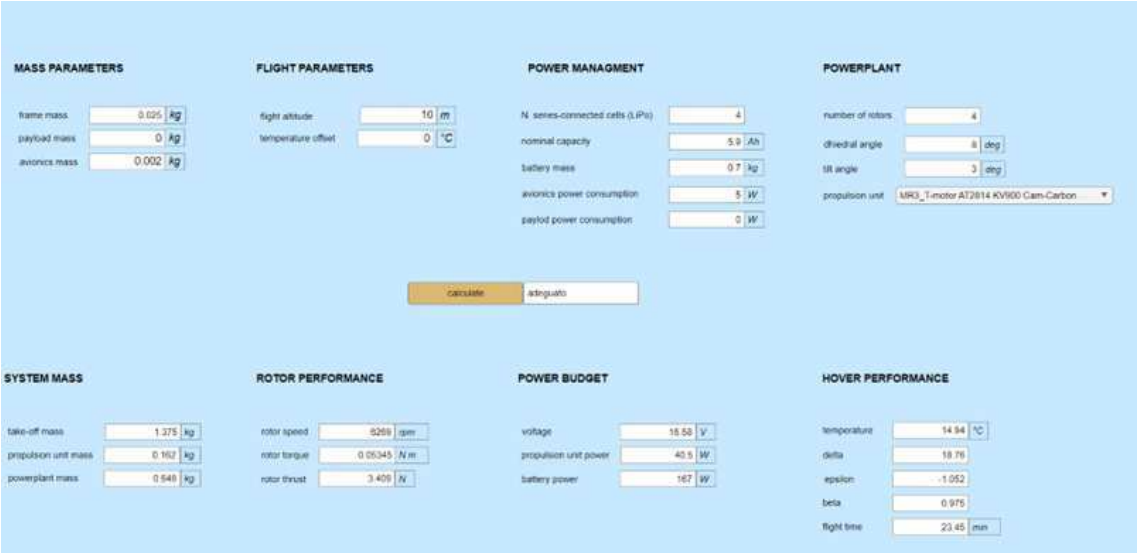


Figura 4.3: calcolo per via sperimentale delle prestazioni di un multirotore di dimensioni ridotte

Considerando tali valori possiamo confrontarli con quelli relativi alla trattazione analitica:

	velocità angolare	coppia	Potenza batteria	Voltaggio	Diametro elica	Passo elica
Analisi aerodinamica	5600 rpm	0.05 Nm	161 W	14.8 V	9 in	4.5 in
Analisi sperimentale	6269 rpm	0.05345 Nm	167 W	16.58 V	10 in	5 in

Si può quindi affermare che i dati sono sufficientemente coerenti in quanto le variazioni presenti sono da imputarsi alle differenti geometrie e caratteristiche di prestazioni presenti nell'unità propulsiva implementata, rispetto all'unità presa in esame nella trattazione analitica.
Si può infine osservare che, dall'analisi della trazione necessaria confrontata con quella disponibile, l'unità propulsiva scelta risulta adeguata alle condizioni di volo imposte. Ne risulta quindi che il volo è possibile e che la sua durata è stimata di 23,45 minuti.

4.2 Multirotore di medie dimensioni (DJI 51000)

Come secondo caso di studio prendiamo in considerazione un multirotore di dimensioni intermedie. In questo esempio viene preso come tecnologia di riferimento il DJI Spreading Wings S1000.
Possiamo considerare un numero di rotori pari a N=8, ogniuno fornito di un'elica in carbonio pieghevole a due pale che presentano un diametro pari a 15in e un passo pari a 5,2in ($corda_{media}$=0.0175 m e $corda_{75}$=0.019 m).
Per quanto concerne le condizioni di volo possiamo considerare una quota pari a 10m con una configurazione che prevede un angolo di diedro pari a 8deg ed un angolo di tilt pari a 3deg. Si considera un offset di temperatura pari a 0°C.
Per quanto riguarda l'alimentazione vengono implementate un numero di celle N_s=6 di tipologia Li-Po con tensione nominale pari a V_0=14,8 V e capacità C_0=5,9 Ah.
L'unità propulsiva è ottenuta implementando un ESC proprietario da 40A e un motore elettrico brushless 4114-PRO con costante di velocità kv=400rpm/V.
La massa totale al decollo risulta quindi pari a 9,5kg. Si considera inoltre una dissipazione di potenza da parte dell'avionica pari a 5W.

Sapendo che $\frac{\Gamma}{D}$= 0,347 e σ=0,0586 dall'analisi analitica si è in grado di determinare che la coppia fornita dal motore è pari a 0,417Nm con una velocità angolare pari a 5920 rpm. La potenza che deve essere fornita dalla batteria risulta quindi pari a 1 492.3W. il fattore di merito risulta essere f=0,605. [17]

Considerando ora di voler risolvere tale problema mediante l'utilizzo dell'applicazione si ha la necessità di determinare il sistema propulsivo più adatto a rappresentare il drone studiato. Per questo scopo viene scelto l'insieme di dati disponibili nel database di Tyto Robotics che implementa i seguenti sistemi:

- Motore T-Motor AT2814 900KV: viene specificato che tale motore presenta un diametro dell'albero pari a 5mm e un peso complessivo di 107g.
- Elica Graupner Cam-Carbon Z 9X5 23X12: viene specificato che tale elica presenta un diametro di 9in e un passo di 5in.
- ESC Zubax Myxa A2: viene specificato che il peso di tale componente corrisponde a 55g.

In seguito, vengono riportati i dati relativi ai test eseguiti e resi disponibili per l'unità propulsiva precedentemente illustrata facendo riferimento sia a quelli espressi in forma grafica che alla loro versione tabulata.

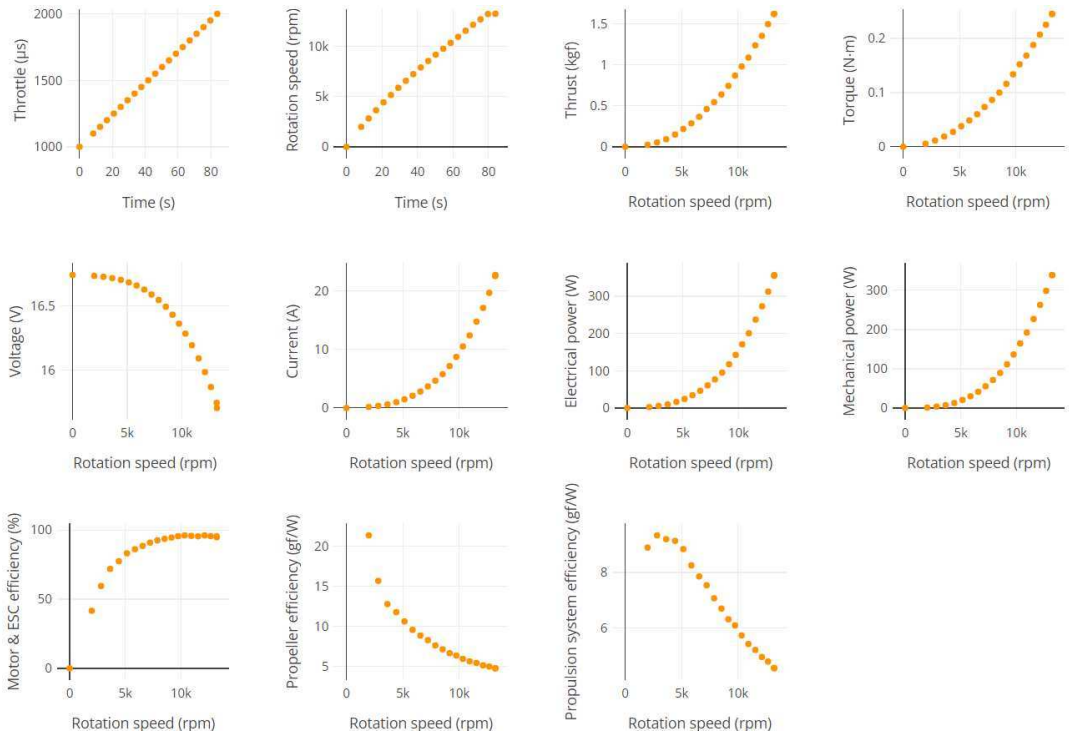


Figura 4.4: grafici dei test relativi al sistema propulsivo di un velivolo di dimensioni medie

Time (s)	Throttle (µs)	Rotation speed (rpm)	Thrust (kgf)	Torque (N·m)	Voltage (V)	Current (A)	Electrical power (W)	Mechanical power (W)	ESC efficiency (%)	Propeller efficiency (gf/W)	system efficiency (gf/W)
0	1000	0	0.0001	-0.0001	16.74	-0.0154	0.2572	0	0		
8.32	1100	1982	0.0242	0.0055	16.73	0.1628	2.725	1.134	41.64	21.37	8.899
12.49	1150	2823	0.0532	0.0115	16.73	0.3409	5.702	3.393	59.5	15.7	9.339
16.68	1200	3635	0.0919	0.0189	16.72	0.5973	9.984	7.179	71.91	12.8	9.202
20.89	1250	4419	0.1495	0.0274	16.7	0.9796	16.36	12.68	77.48	11.8	9.14
25.05	1300	5149	0.2176	0.0379	16.68	1.475	24.6	20.46	83.14	10.64	8.843
29.24	1350	5865	0.2862	0.0486	16.66	2.081	34.66	29.85	86.11	9.589	8.257
33.47	1400	6560	0.366	0.06	16.63	2.801	46.57	41.22	88.52	8.878	7.859
37.65	1450	7222	0.4602	0.0733	16.59	3.679	61.03	55.45	90.85	8.299	7.54
41.86	1500	7883	0.5446	0.0863	16.55	4.655	77.03	71.25	92.5	7.643	7.07
46.07	1550	8533	0.6387	0.0999	16.49	5.78	95.33	89.26	93.64	7.155	6.7
50.28	1600	9151	0.7426	0.1161	16.43	7.159	117.6	111.2	94.56	6.676	6.313
54.49	1650	9742	0.8694	0.1337	16.36	8.722	142.7	136.3	95.55	6.377	6.093
58.69	1700	10327	0.9811	0.1522	16.28	10.51	171.2	164.6	96.13	5.962	5.731
62.89	1750	10926	1.089	0.1681	16.19	12.4	200.8	192.3	95.78	5.66	5.421
67.1	1800	11536	1.237	0.1878	16.09	14.76	237.6	226.8	95.49	5.454	5.208
71.3	1850	12121	1.353	0.2069	15.98	17.1	273.3	262.6	96.11	5.152	4.952
75.5	1900	12664	1.496	0.2252	15.87	19.69	312.4	298.6	95.58	5.011	4.789

Figura 4.5: dati tabulati relativi al sistema propulsivo di un velivolo di dimensioni medie

è ora possibile compilare i campi necessari nell'applicazione seguendo le specifiche precedentemente illustrate ottenendo i seguenti risultati:

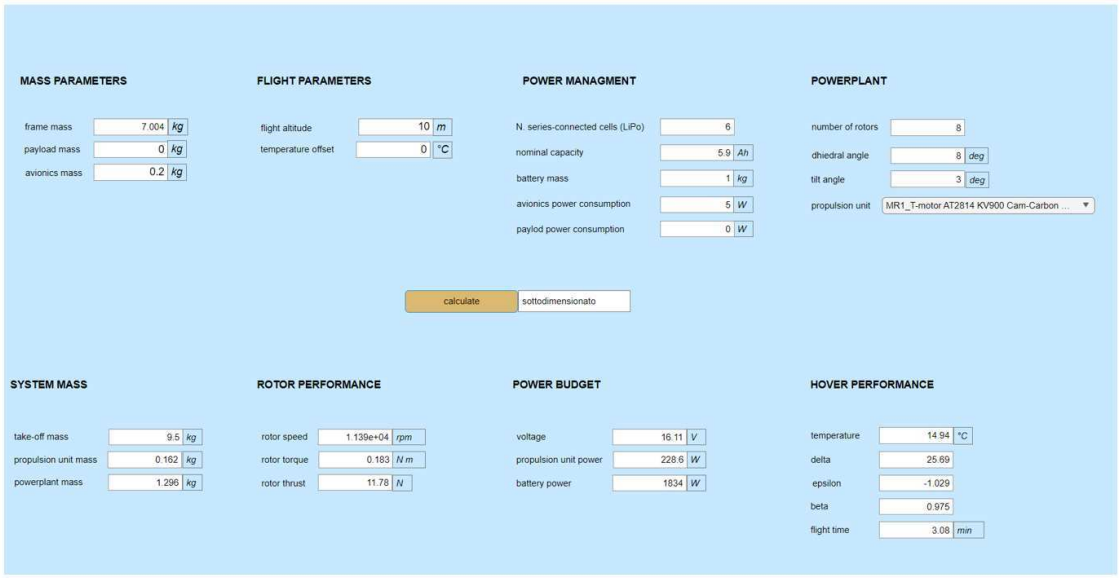


Figura 4.6: calcolo per via sperimentale delle prestazioni di un multirotore di dimensioni medie

Considerando tali valori possiamo confrontarli con quelli relativi alla trattazione aeromeccanica:

	velocità angolare	coppia	Potenza batteria	Voltaggio	Diametro elica	Passo elica
Analisi aerodinamica	5920 rpm	0.417Nm	1 492.3W	14,8 V	15 in	5.2 in
Analisi sperimentale	11390 rpm	0.183Nm	1834 W	16.11 V	9 in	5 in

Dalla comparazione dei dati ora disponibili, è possibile osservare che essi risultano abbastanza discordanti. Questo non è dovuto ad un errore di calcolo bensì da una discrepanza rilevante nella scelta del sistema propulsivo. Si può infatti notare che, nel caso teorico, le dimensioni dell'elica sono decisamente maggiori rispetto a quelle prese in esame nel caso empirico. Ciò è dovuto ad una mancanza di dati, relativi alla velocità di rotazione e voltaggio, in test condotti su unità propulsive con dimensioni più idonee che sono, di conseguenza, state considerate non utilizzabili per l'implementazione del sistema di calcolo.
Infine, possiamo considerare che, per le date condizioni di volo imposte, l'unità propulsiva scelta risulta idonea per permettere il volo ma risulta sottodimensionata e, quindi, sarebbe preferibile operare una scelta di motore, elica ed ESC più efficace.
Il volo in queste condizioni risulta comunque possibile e possiede una durata di 3,08 minuti.

4.3 Multirotore di grandi dimensioni (DJI AGRAS MG-1P)

Come ultimo esempio, si prende in considerazione un multirotore di grandi dimensioni prendendo spunto anche in questo caso ad una tecnologia realmente esistente ovvero l'AGRAS MG-1P, progettato per attività agricola.
Possiamo considerare un numero di rotori pari a N=8, ognuno fornito di un'elica in carbonio pieghevole a due pale che presentano un diametro pari a 21in e un passo pari a 7in ($corda_{media}$=0,029 m e $corda_{75}$=0,021 m).
Per quanto concerne le condizioni di volo, possiamo considerare una quota pari a 10m con una configurazione che prevede un angolo di diedro pari a 0deg ed un angolo di tilt pari a 3deg. Si considera un offset di temperatura pari a 0°C.
Per quanto riguarda l'alimentazione, vengono implementate un numero di celle N_s=6 di tipologia Li-Po MG-12000P con tensione nominale V_0=44.4V e capacità C_0=12Ah.
L'unità propulsiva è ottenuta implementando un sistema integrato DJI composto da un ESC da 25A e un motore elettrico brushless 6010.
La massa al decollo totale risulta quindi pari a 22,5kg. Si considera inoltre una dissipazione di potenza da parte dell'avionica pari a 10W e un consumo di energia da parte del payload pari a 40W.

Eseguando l'analisi aeromeccanica si è in grado di determinare che la coppia fornita dal motore è pari a 1,16Nm. La potenza che deve essere fornita dalla batteria risulta quindi pari a 3 114,3W. Il coefficiente di merito risulta essere pari a f=0,635 [17]

Considerando ora di voler risolvere tale problema mediante l'utilizzo dell'applicazione si ha in primo luogo la necessità di determinare il sistema propulsivo più adatto a rappresentare il drone studiato. Per questo scopo viene scelto l'insieme di dati disponibili nel database di Tyto Robotics che implementa i seguenti sistemi:

- Motore Scorpion Power System LTD A-5025-220kv: viene specificato che il diametro dell'albero è pari a 8 mm, possiede 14 poli magnetici e ha una massa pari a 523g.
- elica Xoar PJN - 21 x 8: viene specificato che il diametro dell'elica è pari a 21in, il passo di 8in ed una massa di 69g.
- ESC Scorpion Power System LTD Tribunus II 12-80A SBEC: viene specificato che esso supporta una corrente massima di 120A e possiede una massa di 144g.

In seguito, vengono riportati i dati relativi ai test eseguiti e resi disponibili per l'unità propulsiva precedentemente illustrata facendo riferimento sia a quelli espressi in forma grafica che alla loro versione tabulata.

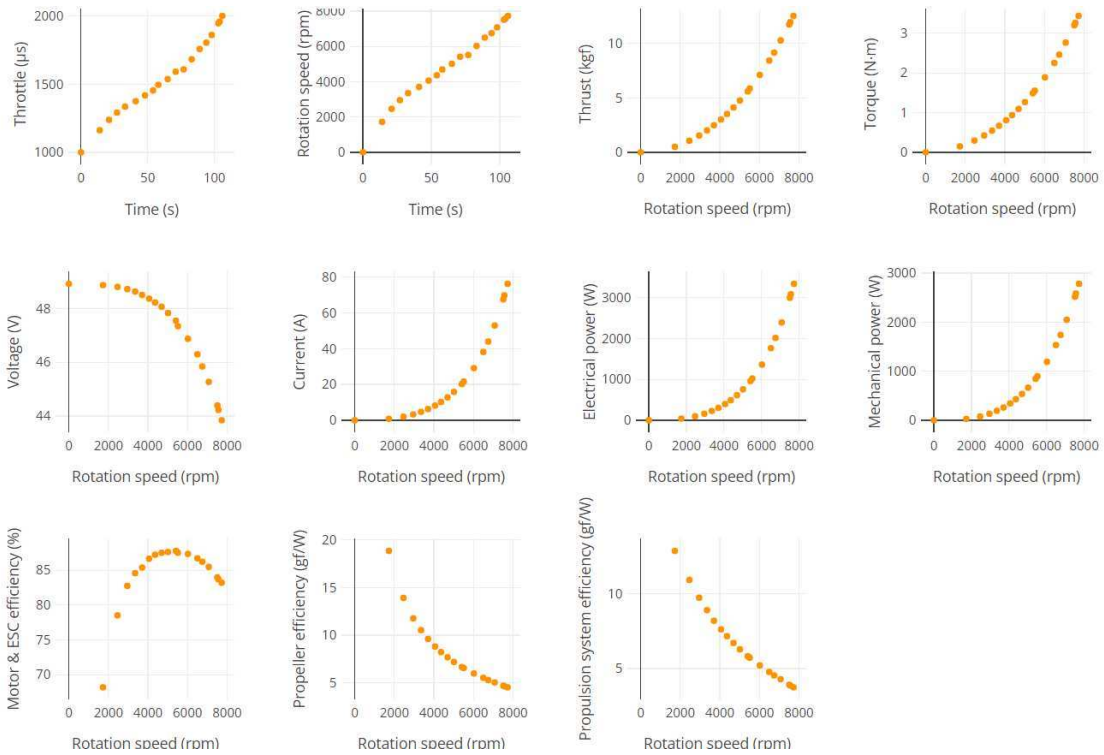


Figura 4.7: grafici dei test relativi al sistema propulsivo di un velivolo di grandi dimensioni

Time (s)	Throttle (µs)	speed (rpm)	Thrust (kgf)	Torque (N·m)	Voltage (V)	Current (A)	power (W)	Mechanical power (W)	efficiency (%)	efficiency (gf/W)	efficiency (gf/W)
0	1000	0	0.0007	0	48.92	0	0	0			
14	1162	1727	0.5111	0.1499	48.87	0.8137	39.76	27.11	68.18	18.85	12.85
21	1238	2462	1.061	0.2962	48.8	1.992	97.23	76.34	78.52	13.9	10.92
27	1291	2955	1.547	0.425	48.72	3.262	158.9	131.5	82.75	11.76	9.732
33	1335	3352	2.021	0.5472	48.63	4.67	227.1	192.1	84.58	10.52	8.9
41	1375	3704	2.489	0.6686	48.5	6.263	303.8	259.4	85.39	9.597	8.195
48	1417	4061	3.021	0.8079	48.37	8.197	396.5	343.6	86.65	8.793	7.619
54	1454	4364	3.524	0.9383	48.22	10.2	491.7	428.8	87.22	8.217	7.167
58	1495	4693	4.121	1.093	48.07	12.77	614	537.3	87.51	7.67	6.712
65	1537	5016	4.766	1.265	47.83	15.85	758	664.2	87.62	7.175	6.287
71	1591	5415	5.598	1.487	47.55	20.2	960.6	843.2	87.78	6.639	5.827
77	1608	5513	5.864	1.552	47.34	21.61	1023	895.6	87.52	6.547	5.73
83	1682	6024	7.11	1.888	46.87	29.09	1364	1191	87.35	5.969	5.214
89	1757	6500	8.439	2.25	46.3	38.15	1766	1531	86.71	5.51	4.778
94	1804	6747	9.168	2.46	45.84	43.97	2016	1738	86.22	5.275	4.548
98	1860	7077	10.29	2.763	45.27	52.9	2395	2047	85.48	5.025	4.296
103	1946	7513	11.76	3.2	44.39	67.53	2998	2517	83.97	4.67	3.921
104	1958	7564	11.97	3.261	44.22	69.78	3086	2583	83.69	4.636	3.88

Figura 4.8: dati tabulati relativi al sistema propulsivo di un velivolo di grandi dimensioni

è ora possibile compilare i campi necessari nell'applicazione seguendo le specifiche precedentemente illustrate ottenendo i seguenti risultati:

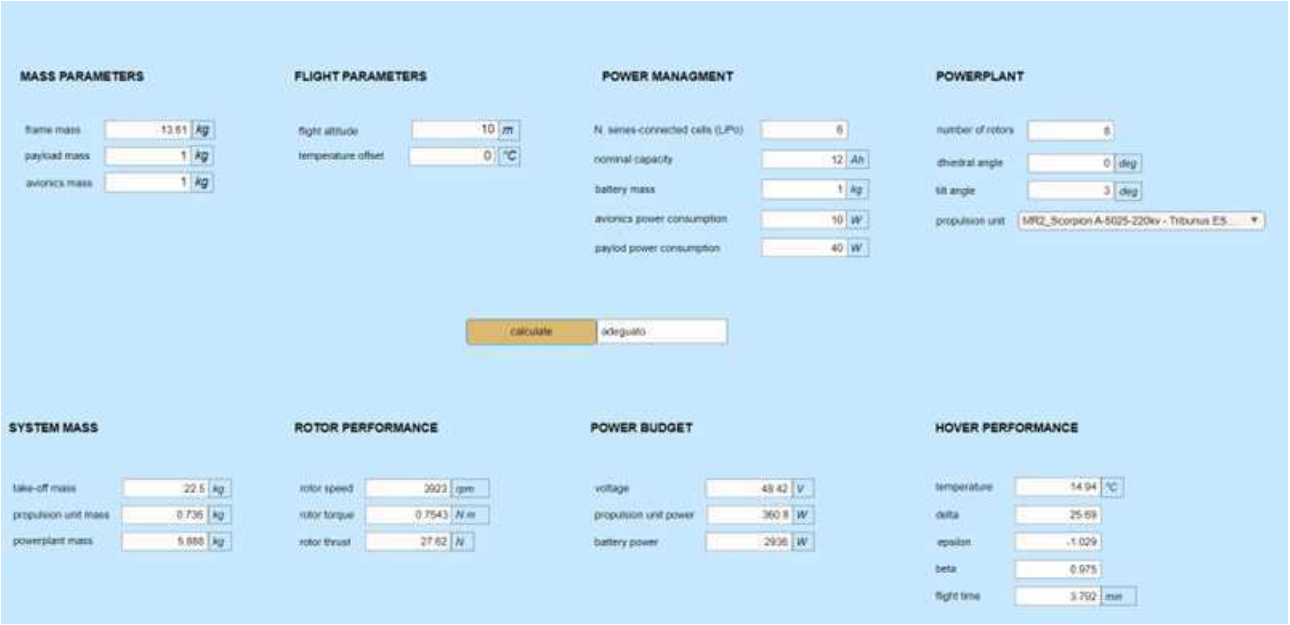


Figura 4.9: calcolo per via sperimentale delle prestazioni di un multirotore di grandi dimensioni

Considerando tali valori possiamo confrontarli con quelli relativi alla trattazione analitica:

	coppia	Potenza batteria	Voltaggio	Diametro elica	Passo elica
Analisi aerodinamica	1.16Nm	3 114.3W	44.4	21 in	7 in
Analisi sperimentale	0.7543 N m	2936 W	48.42	21 in	8 in

Analizzando i dati trovati possiamo vedere che quelli relativi all'utilizzo dell'applicazione differiscono da quelli trovati mediante analisi aeromeccanica. Tali variazioni però non sono sufficientemente importanti da considerare il modello implementato nel seguente lavoro di tesi come non idoneo a rappresentare la realtà.
Considerando ora i risultati ottenuti per via sperimentale, si può osservare che la scelta dell'unità propulsiva appartenente al database online sia in grado di soddisfare le condizioni operative imposte. Se ne deduce, infatti, che il volo risulta in tale caso possibile e si può stimarne la durata, facendo riferimento al tempo di scarica della batteria, pari a 3.792 minuti.

4.4 Osservazioni

Considerando gli esempi appena descritti, se ne può dedurre che l'implementazione dell'applicazione relativa al calcolo delle prestazioni di un multirotore risulta affidabile per una stima della potenza necessaria e il tempo di volo.
Le variazioni di valori, rispetto all'analisi aerodinamica presa come esempio, sono presenti ma non sono imputabili alla differenza dell'unità propulsiva presa in esame.

Un limite di tale tecnologia, infatti, è la possibilità di prendere in considerazione solo unità propulsive che presentano un'associazione di motore elettrico, elica ed ESC precedentemente testate, e di cui si hanno a disposizione, tabulati, tutti i dati necessari.
Come successo durante lo studio dei tre casi presi in esame, si è notato che le misure dei componenti dello studio teorico non corrispondevano a quelle implementate nello studio mediante l'uso dell'applicazione, in quanto non erano a disposizione test relativi a componenti delle misure desiderate.

Un'altra osservazione che si può fare, riguardo lo studio delle prestazioni di un multirotore per via sperimentale, è la riduzione del tempo relativo alle fasi di calcolo e la possibilità di stabilire se un dato sistema propulsivo sia adeguato o meno per effettuare il volo nelle condizioni richieste.
In alternativa, è possibile stabilire quali condizioni operative possono essere raggiunte dato uno specifico sistema propulsivo. Questo può essere stabilito ad esempio variando la quota di volo o gli angoli di tilt e diedro.

Un altro vantaggio dell'analizzare un sistema mediante tale interfaccia, è quello di poter cambiare i parametri relativi alla batteria implementata per fornire energia al sistema.
Cambiando i dati relativi a capacità e massa si ha la possibilità di analizzare differenti soluzioni progettuali, spaziando tra le differenti tecnologie di batterie più in uso nel settore aeronautico.
In conclusione, quindi, da tale confronto è stato possibile mettere in luce i differenti vantaggi che l'applicazione, sviluppata durante il lavoro di tesi, offre ed è stato possibile dimostrare che offre una stima sufficientemente buona delle caratteristiche fondamentali dei multirotori. Questo software risulta così adatto e utile durante la progettazione preliminare di tale tecnologia.

Capitolo 5

Conclusioni

Lo studio effettuato durante il lavoro di tesi aveva come finalità l'implementazione di un software. Questo doveva possedere le opportune funzionalità, in termini di capacità di calcolo e facilità di utilizzo, per garantire un supporto durante le fasi preliminari di progettazione di un multirotore.

A tale scopo, si è dovuto, in primo luogo, dimostrare la teoria che descrive il comportamento di tale tecnologia, ponendo l'attenzione sulla condizione di hover, illustrando le varie funzioni che determinano il volo. Si è infatti descritto il processo analitico per l'ottenimento di una stima di potenza elettrica della batteria e tempo di volo del multirotore. Da queste considerazioni si è stabilito il punto di partenza per l'implementazione dell'applicazione, mettendo in luce le differenti grandezze che dovevano fungere da input o da output.

Successivamente, si è ricorsi all'utilizzo di App Designer per la scrittura e il design dell'applicazione, illustrandone le varie componenti e il loro funzionamento. Si è fatto riferimento alle diverse problematiche incorse durante la fase di programmazione, come ad esempio, la necessità di dover rendere i dati sperimentali disponibili in un formato compatibile con MATLAB oppure, considerando i file relativi ai test condotti, la loro mancanza di dati e, di conseguenza, l'impossibilità di utilizzarli. In secondo luogo, è stato illustrato il procedimento logico svolto dal software per ottenere gli output desiderati, mettendo in luce le varie sezioni che compongono il codice e le loro diverse funzionalità. Si è così potuto rendere nota la possibilità, che offre App Designer, di poter implementare un'applicazione in grado di svolgere differenti funzioni operando solamente attraverso l'interfaccia grafica. Questo permette di semplificare l'utilizzo del software e renderlo più accessibile.

Infine, è stato necessario dimostrare che l'applicazione sviluppata possa essere effettivamente applicabile nello studio di multirotori, ovvero che i risultati ottenuti mediante il suo utilizzo descrivano in modo sufficientemente accurato la realtà.
Per fare ciò sono stati presi come riferimento gli studi analitici condotti in relazione a tre differenti multirotori, che differiscono tra loro per dimensione. Conoscendone i requisiti e le relative condizioni di volo, è stato possibile studiare tali tecnologie mediante l'utilizzo del software implementato. I risultati ottenuti hanno permesso di dimostrare che l'applicazione è in grado di offrire delle stime di potenza elettrica e tempo di volo congrue con quelle effettivamente richieste dal caso reale.
A tal proposito, si deve mettere in luce un limite nell'utilizzo del software, legato alla scelta del sistema propulsivo. Nell'inserimento degli input, infatti, si ha la possibilità di scegliere l'unità propulsiva da implementare nel multirotore scegliendo tra quelle disponibili, ovvero precedentemente

testate in laboratorio. Se ne deduce quindi che nella scelta del motore elettrico, eliche ed ESC ci si debba limitare a tecnologie di cui siano disponibili i dati relativi ai test condotti e non alla totalità di componenti presenti sul mercato. La possibilità, offerta da App Designer, di poter modificare il codice dell'applicazione però permette di poter ampliare il database di unità propulsive disponibili, avendo così la possibilità di aggiungere i dati, ottenuti precedentemente per via sperimentale, relativi alla tecnologia di interesse.

Quest'ultima considerazione apre all'idea di poter ampliare le informazioni contenute nel software ma anche di implementare opportune parti di codice per aumentarne le funzionalità.
Nello studio condotto durante il lavoro di tesi, infatti, si è fatto riferimento a multirotori in condizione di hover. Ampliando il codice del software però, è possibile estenderne le capacità di calcolo anche ad altre applicazioni. Ad esempio, è possibile prendere in esame velivoli elettrici differenti come elicotteri, velivoli ad ala fissa o anche mezzi su ruote.
In secondo luogo, è possibile implementare opportune formule analitiche in grado di tener conto, nel calcolo delle differenti grandezze di interesse, anche di effetti secondari, considerati trascurabili nella trattazione. In questo modo risulta possibile aumentare il grado di precisione dei risultati ottenuti.

In conclusione, si può affermare che i risultati ricercati tramite il lavoro di tesi siano stati raggiunti, in quanto si è dimostrato possibile implementare un software in grado di offrire delle stime di potenza e tempo di volo sufficientemente adeguate. Questo può essere solo il punto di inizio per una possibile estensione dell'applicazione ad altri ambiti di interesse.

Appendice

```
classdef app4 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        DropDown                        matlab.ui.control.DropDown
        DropDownLabel                   matlab.ui.control.Label
        unit3_6                         matlab.ui.control.EditField
        flighttimeEditField             matlab.ui.control.NumericEditField
        betaEditField                   matlab.ui.control.NumericEditField
        betaEditFieldLabel              matlab.ui.control.Label
        flighttimeEditFieldLabel        matlab.ui.control.Label
        pay_pow_cons_unit_5             matlab.ui.control.EditField
        batterypowerEditField           matlab.ui.control.NumericEditField
        batterypowerEditFieldLabel      matlab.ui.control.Label
        unit1_2                         matlab.ui.control.EditField
        rotorthrustEditField            matlab.ui.control.NumericEditField
        rotorthrustEditFieldLabel       matlab.ui.control.Label
        unit5                           matlab.ui.control.EditField
        powerplantmassEditField         matlab.ui.control.NumericEditField
        Label_2                         matlab.ui.control.Label
        epsilonEditField                matlab.ui.control.NumericEditField
        deltaEditField                  matlab.ui.control.NumericEditField
        deltaEditFieldLabel             matlab.ui.control.Label
        epsilonEditFieldLabel           matlab.ui.control.Label
        pay_pow_cons_unit_2             matlab.ui.control.EditField
        unit3_5                         matlab.ui.control.EditField
        propulsionunitpowerEditField    matlab.ui.control.NumericEditField
        voltageEditField                matlab.ui.control.NumericEditField
        voltageEditFieldLabel           matlab.ui.control.Label
        propulsionunitpowerEditFieldLabel  matlab.ui.control.Label
        pay_pow_cons_unit_4             matlab.ui.control.EditField
        rotortorqueEditField            matlab.ui.control.NumericEditField
        rotortorqueEditFieldLabel       matlab.ui.control.Label
        unit4                           matlab.ui.control.EditField
        propulsionunitmassEditField     matlab.ui.control.NumericEditField
        propulsionunitmassEditFieldLabel  matlab.ui.control.Label
        unit_temp                       matlab.ui.control.EditField
        temperatureEditField            matlab.ui.control.NumericEditField
        temperatureEditFieldLabel       matlab.ui.control.Label
        pay_pow_cons_unit_3             matlab.ui.control.EditField
        rotorspeedEditField             matlab.ui.control.NumericEditField
        rotorspeedEditFieldLabel        matlab.ui.control.Label
        unit6                           matlab.ui.control.EditField
        takeoffmassEditField            matlab.ui.control.NumericEditField
        takeoffmassLabel                matlab.ui.control.Label
        HOVERPERFORMANCELabel           matlab.ui.control.Label
        POWERBUDGETLabel                matlab.ui.control.Label
        ROTORPERFORMANCELabel           matlab.ui.control.Label
```

```
    SYSTEMMASSLabel                 matlab.ui.control.Label
    warning                         matlab.ui.control.EditField
    calculateButton                 matlab.ui.control.Button
    pay_pow_cons_unit               matlab.ui.control.EditField
    paylodpowerconsumptionEditField  matlab.ui.control.NumericEditField
    paylodpowerconsumptionEditFieldLabel  matlab.ui.control.Label
    propulsionunitDropDown          matlab.ui.control.DropDown
    propulsionunitDropDownLabel     matlab.ui.control.Label
    avionic_power_consumption_unit  matlab.ui.control.EditField
    avionicspowerconsumptionEditField  matlab.ui.control.NumericEditField
    avionicspowerconsumptionEditFieldLabel  matlab.ui.control.Label
    unit3                           matlab.ui.control.EditField
    avionicsmassEditField           matlab.ui.control.NumericEditField
    avionicsmassEditFieldLabel      matlab.ui.control.Label
    unit3_3                         matlab.ui.control.EditField
    tiltangleEditField              matlab.ui.control.NumericEditField
    tiltangleEditFieldLabel         matlab.ui.control.Label
    unit1_3                         matlab.ui.control.EditField
    batterymassEditField            matlab.ui.control.NumericEditField
    batterymassEditField_2Label     matlab.ui.control.Label
    unit2                           matlab.ui.control.EditField
    payloadmassEditField            matlab.ui.control.NumericEditField
    payloadmassEditFieldLabel       matlab.ui.control.Label
    unit3_4                         matlab.ui.control.EditField
    dhiedralangleEditField          matlab.ui.control.NumericEditField
    dhiedralangleEditFieldLabel     matlab.ui.control.Label
    unit3_2                         matlab.ui.control.EditField
    nominalcapacityEditField        matlab.ui.control.NumericEditField
    nominalcapacityEditFieldLabel   matlab.ui.control.Label
    unit_altitude_2                 matlab.ui.control.EditField
    temperatureoffsetEditField      matlab.ui.control.NumericEditField
    temperatureoffsetEditFieldLabel  matlab.ui.control.Label
    unit1                           matlab.ui.control.EditField
    framemassEditField              matlab.ui.control.NumericEditField
    framemassEditFieldLabel         matlab.ui.control.Label
    numberofrotorsEditField         matlab.ui.control.NumericEditField
    POWERPLANTLabel                 matlab.ui.control.Label
    numberofrotorsEditFieldLabel    matlab.ui.control.Label
    NseriesconnectedcellsLiPoEditField  matlab.ui.control.NumericEditField
    POWERMANAGMENTLabel             matlab.ui.control.Label
    NseriesconnectedLabel           matlab.ui.control.Label
    unit_altitude                   matlab.ui.control.EditField
    flightaltitudeEditField         matlab.ui.control.NumericEditField
    flightaltitudeEditFieldLabel    matlab.ui.control.Label
    FLIGHTPARAMETERSLabel           matlab.ui.control.Label
    MASSPARAMETERSLabel             matlab.ui.control.Label
end

% Callbacks that handle component events
methods (Access = private)

    % Button pushed function: calculateButton
    function calculateButtonPushed(app, event)

%scelta massa sistema propulsivo

        switch app.propulsionunitDropDown.Value


        case 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop 10X5 25X12_Zubax Myxa A2'
                app.propulsionunitmassEditField.Value=0.162;
        case 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop 10X6-3 25X16_ Zubax Myxa A2'
                app.propulsionunitmassEditField.Value=0.162;
```

```
            case 'T-Motor AT2814 900KV_ Graupner Cam-Carbon Z 9X5 23X12_ Zubax Myxa A2'
                     app.propulsionunitmassEditField.Value=0.162;
            case 'T-Motor AT2814 900KV_Graupner CAM Folding Prop 10X6 25X15_Zubax Myxa A2'
                     app.propulsionunitmassEditField.Value=0.162;
            case 'Scorpion Power System LTD IM-8012-115kv_ Scorpion Power System LTD PF 28_
Scorpion Power System LTD Legatus 12-60A FOC'
                    app.propulsionunitmassEditField.Value=0.605;
            case 'MR1_T-motor AT2814 KV900 Cam-Carbon Z 9X5 23X12 test test data'
                    app.propulsionunitmassEditField.Value=0.162;
            case 'MR2_Scorpion A-5025-220kv - Tribunus ESC - Xoar 21x8E test data'
                    app.propulsionunitmassEditField.Value=0.736;
            case 'MR3_T-motor AT2814 KV900 Cam-Carbon Z 10X5 25X12 test test data'
                    app.propulsionunitmassEditField.Value=0.162;

            end


            %calcoli temperatura esterna
            h=app.flightaltitudeEditField.Value;
            dt=app.temperatureoffsetEditField.Value;

            if h<11000
                temp1=288.15-0.0065*h;
                temp=(temp1+dt)-273.15;
            else
                temp=(216.65+dt)-273.15;
            end



    %calcoli per batteria

            Ns=app.NseriesconnectedcellsLiPoEditField.Value;
            C0=app.nominalcapacityEditField.Value;
            battery_mass=app.batterymassEditField.Value;

            del0=-0.1067*(Ns^3)+0.8960*(Ns^2)+2.488*Ns+0.6299;
            eps0=2.917*(10^(-4))*(Ns^3)-1.375*(10^(-3))*(Ns^2)+3.083*(10^(-3))*Ns-1.041;

            del=del0*(1-0.0046*(temp-23));
            eps=eps0*(1-0.0024*(temp-23));
            beta=0.9664*(1-0.0011*(temp-23));
            Cf=0.8*C0;

            app.epsilonEditField.Value=eps;
            app.betaEditField.Value=beta;
            app.deltaEditField.Value=del;

    %calcoli per massa
            frame_mass=app.framemassEditField.Value;
            avionic_mass=app.avionicsmassEditField.Value;
            payload_mass=app.payloadmassEditField.Value;
            number_of_rotors=app.numberofrotorsEditField.Value;
            propunit_mass=app.propulsionunitmassEditField.Value;

            powerplant_mass=number_of_rotors*propunit_mass;
            total_mass=payload_mass+frame_mass+avionic_mass+powerplant_mass+battery_mass;


    %calcoli T
            tilt=app.tiltangleEditField.Value;
            diedral=app.dhiedralangleEditField.Value;

            T=(total_mass/number_of_rotors)*9.80665;
```

```
T_rot=T/(cos(tilt*(3.14/180))*cos(diedral*(3.14/180)));

total_thrust=T_rot*number_of_rotors;




 %scelta sistema propulsivo

 switch app.propulsionunitDropDown.Value


 case 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop 10X5 25X12_Zubax Myxa A2'
        load T-MotorAT2814900KV_GraupnerCAMFoldingProp10X525X12_ZubaxMyxaA2.mat;

 case 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop 10X6-3 25X16_ Zubax Myxa A2'
        load T-MotorAT2814900KV_GraupnerCAMFoldingProp10X6-325X16_ZubaxMyxaA2.mat;

 case 'T-Motor AT2814 900KV_ Graupner Cam-Carbon Z 9X5 23X12_ Zubax Myxa A2'
        load T-MotorAT2814900KV_GraupnerCam-CarbonZ9X523X12_ZubaxMyxaA2.mat;

 case 'T-Motor AT2814 900KV_Graupner CAM Folding Prop 10X6 25X15_Zubax Myxa A2'
        load T-MotorAT2814900KV_GraupnerCAMFoldingProp10X625X15_ZubaxMyxaA2.mat;

 case 'Scorpion Power System LTD IM-8012-115kv_ Scorpion Power System LTD PF 28_
Scorpion Power System LTD Legatus 12-60A FOC'
        load ScorpionPowerSystemLTDIM-8012-
115kv_ScorpionPowerSystemLTDPF28_ScorpionPowerSystemLTDLegatus12-60AFOC.mat;

     case 'MR1_T-motor AT2814 KV900 Cam-Carbon Z 9X5 23X12 test test data'
        load MR1_T-motorAT2814KV900Cam-CarbonZ9X523X12testtestdata.mat

     case 'MR2_Scorpion A-5025-220kv - Tribunus ESC - Xoar 21x8E test data'
        load MR2_ScorpionA-5025-220kv-TribunusESC-Xoar21x8Etestdata.mat

     case 'MR3_T-motor AT2814 KV900 Cam-Carbon Z 10X5 25X12 test test data'
        load MR3_T-motorAT2814KV900Cam-CarbonZ10X525X12testtestdata.mat

 end

        thrust1=tabella(:,2);
        thrust=thrust1*9.80665;
        a=max(thrust);
        asicurezza=a/2;

    if   T_rot<=asicurezza

        pe=tabella(:,5);
        Vv=tabella(:,4);
        rotspeed=tabella(:,1);
        tor=tabella(:,3);

        Pel=interp1(thrust1,pe,T_rot/9.80665);
        V=interp1(thrust1,Vv,T_rot/9.80665);
        rot=interp1(thrust1,rotspeed,T_rot/9.80665);
        trq=interp1(thrust1,tor,T_rot/9.80665);

        app.propulsionunitpowerEditField.Value=Pel;
        app.voltageEditField.Value=V;
        app.rotorspeedEditField.Value=rot;
        app.rotortorqueEditField.Value=trq;
        app.warning.Value='adeguato';

    elseif  T_rot<=a
```

```
            pe=tabella(:,5);
            Vv=tabella(:,4);
            rotspeed=tabella(:,1);
            tor=tabella(:,3);

            Pel=interp1(thrust1,pe,T_rot/9.80665);
            V=interp1(thrust1,Vv,T_rot/9.80665);
            rot=interp1(thrust1,rotspeed,T_rot/9.80665);
            trq=interp1(thrust1,tor,T_rot/9.80665);

            app.propulsionunitpowerEditField.Value=Pel;
            app.voltageEditField.Value=V;
            app.rotorspeedEditField.Value=rot;
            app.rotortorqueEditField.Value=trq;
            app.warning.Value='sottodimensionato';


        else
            Pel=0;
            V=0;
            rot=0;
            trq=0;

            app.propulsionunitpowerEditField.Value=Pel;
            app.voltageEditField.Value=V;
            app.rotorspeedEditField.Value=rot;
            app.rotortorqueEditField.Value=trq;
            app.warning.Value='errore';


        end


    Pav=app.avionicspowerconsumptionEditField.Value;
    Ppay=app.paylodpowerconsumptionEditField.Value;

    total_pe=Pel*number_of_rotors+Pav;
    Pb=(number_of_rotors*Pel)+Pav+Ppay;

    tscarica=(del*(Pb^eps)*(Cf^beta))*60;

    %mostrare risultati

    app.takeoffmassEditField.Value=total_mass;
    app.powerplantmassEditField.Value=powerplant_mass;

    app.rotorthrustEditField.Value=T_rot;

    app.batterypowerEditField.Value=Pb;

    app.flighttimeEditField.Value=tscarica;
    app.temperatureEditField.Value=temp;


end

% Value changed function: propulsionunitDropDown
function propulsionunitDropDownValueChanged(app, event)



end
```

```
        % Callback function
        function unitofmeasureButtonGroupSelectionChanged(app, event)
          % unit of measure



        end

        % Drop down opening function: propulsionunitDropDown
        function propulsionunitDropDownOpening(app, event)


        end
    end
% Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [0.7804 0.9098 1];
            app.UIFigure.Position = [100 100 1544 777];
            app.UIFigure.Name = 'MATLAB App';

            % Create MASSPARAMETERSLabel
            app.MASSPARAMETERSLabel = uilabel(app.UIFigure);
            app.MASSPARAMETERSLabel.FontSize = 14;
            app.MASSPARAMETERSLabel.FontWeight = 'bold';
            app.MASSPARAMETERSLabel.Position = [28 657 146 22];
            app.MASSPARAMETERSLabel.Text = 'MASS PARAMETERS';

            % Create FLIGHTPARAMETERSLabel
            app.FLIGHTPARAMETERSLabel = uilabel(app.UIFigure);
            app.FLIGHTPARAMETERSLabel.FontSize = 14;
            app.FLIGHTPARAMETERSLabel.FontWeight = 'bold';
            app.FLIGHTPARAMETERSLabel.Position = [351 657 156 22];
            app.FLIGHTPARAMETERSLabel.Text = 'FLIGHT PARAMETERS';

            % Create flightaltitudeEditFieldLabel
            app.flightaltitudeEditFieldLabel = uilabel(app.UIFigure);
            app.flightaltitudeEditFieldLabel.HorizontalAlignment = 'right';
            app.flightaltitudeEditFieldLabel.Position = [351 595 76 22];
            app.flightaltitudeEditFieldLabel.Text = 'flight altitude ';

            % Create flightaltitudeEditField
            app.flightaltitudeEditField = uieditfield(app.UIFigure, 'numeric');
            app.flightaltitudeEditField.FontSize = 14;
            app.flightaltitudeEditField.Position = [489 597 100 22];

            % Create unit_altitude
            app.unit_altitude = uieditfield(app.UIFigure, 'text');
            app.unit_altitude.FontSize = 14;
            app.unit_altitude.FontAngle = 'italic';
            app.unit_altitude.BackgroundColor = [0.7804 0.9098 1];
            app.unit_altitude.Position = [588 596 26 23];
            app.unit_altitude.Value = 'm';

            % Create NseriesconnectedLabel
            app.NseriesconnectedLabel = uilabel(app.UIFigure);
            app.NseriesconnectedLabel.HorizontalAlignment = 'right';
            app.NseriesconnectedLabel.Position = [701 597 174 22];
            app.NseriesconnectedLabel.Text = 'N. series-connected cells (LiPo)';
```

```
% Create POWERMANAGMENTLabel
app.POWERMANAGMENTLabel = uilabel(app.UIFigure);
app.POWERMANAGMENTLabel.FontSize = 14;
app.POWERMANAGMENTLabel.FontWeight = 'bold';
app.POWERMANAGMENTLabel.Position = [712 657 158 22];
app.POWERMANAGMENTLabel.Text = 'POWER MANAGMENT ';

% Create NseriesconnectedcellsLiPoEditField
app.NseriesconnectedcellsLiPoEditField = uieditfield(app.UIFigure, 'numeric');
app.NseriesconnectedcellsLiPoEditField.Position = [898 597 100 22];
app.NseriesconnectedcellsLiPoEditField.Value = 6;

% Create numberofrotorsEditFieldLabel
app.numberofrotorsEditFieldLabel = uilabel(app.UIFigure);
app.numberofrotorsEditFieldLabel.HorizontalAlignment = 'right';
app.numberofrotorsEditFieldLabel.Position = [1102 597 93 22];
app.numberofrotorsEditFieldLabel.Text = 'number of rotors';

% Create POWERPLANTLabel
app.POWERPLANTLabel = uilabel(app.UIFigure);
app.POWERPLANTLabel.FontSize = 14;
app.POWERPLANTLabel.FontWeight = 'bold';
app.POWERPLANTLabel.Position = [1106 657 105 22];
app.POWERPLANTLabel.Text = 'POWERPLANT';

% Create numberofrotorsEditField
app.numberofrotorsEditField = uieditfield(app.UIFigure, 'numeric');
app.numberofrotorsEditField.Position = [1213 596 100 22];
app.numberofrotorsEditField.Value = 8;

% Create framemassEditFieldLabel
app.framemassEditFieldLabel = uilabel(app.UIFigure);
app.framemassEditFieldLabel.HorizontalAlignment = 'right';
app.framemassEditFieldLabel.Position = [29 597 68 22];
app.framemassEditFieldLabel.Text = 'frame mass';

% Create framemassEditField
app.framemassEditField = uieditfield(app.UIFigure, 'numeric');
app.framemassEditField.Position = [126 597 100 22];
app.framemassEditField.Value = 7.204;

% Create unit1
app.unit1 = uieditfield(app.UIFigure, 'text');
app.unit1.FontSize = 14;
app.unit1.FontAngle = 'italic';
app.unit1.BackgroundColor = [0.7804 0.9098 1];
app.unit1.Position = [226 597 26 22];
app.unit1.Value = 'kg';

% Create temperatureoffsetEditFieldLabel
app.temperatureoffsetEditFieldLabel = uilabel(app.UIFigure);
app.temperatureoffsetEditFieldLabel.HorizontalAlignment = 'right';
app.temperatureoffsetEditFieldLabel.Position = [351 567 102 22];
app.temperatureoffsetEditFieldLabel.Text = 'temperature offset';

% Create temperatureoffsetEditField
app.temperatureoffsetEditField = uieditfield(app.UIFigure, 'numeric');
app.temperatureoffsetEditField.FontSize = 14;
app.temperatureoffsetEditField.Position = [485 567 100 22];

% Create unit_altitude_2
app.unit_altitude_2 = uieditfield(app.UIFigure, 'text');
app.unit_altitude_2.FontSize = 14;
app.unit_altitude_2.FontAngle = 'italic';
app.unit_altitude_2.BackgroundColor = [0.7804 0.9098 1];
```

```
app.unit_altitude_2.Position = [588 566 26 23];
app.unit_altitude_2.Value = '°C';

% Create nominalcapacityEditFieldLabel
app.nominalcapacityEditFieldLabel = uilabel(app.UIFigure);
app.nominalcapacityEditFieldLabel.HorizontalAlignment = 'right';
app.nominalcapacityEditFieldLabel.Position = [701 563 94 22];
app.nominalcapacityEditFieldLabel.Text = 'nominal capacity';

% Create nominalcapacityEditField
app.nominalcapacityEditField = uieditfield(app.UIFigure, 'numeric');
app.nominalcapacityEditField.Position = [898 563 100 22];
app.nominalcapacityEditField.Value = 5.9;

% Create unit3_2
app.unit3_2 = uieditfield(app.UIFigure, 'text');
app.unit3_2.FontAngle = 'italic';
app.unit3_2.BackgroundColor = [0.7804 0.9098 1];
app.unit3_2.Position = [997 563 26 22];
app.unit3_2.Value = 'Ah';

% Create dhiedralangleEditFieldLabel
app.dhiedralangleEditFieldLabel = uilabel(app.UIFigure);
app.dhiedralangleEditFieldLabel.HorizontalAlignment = 'right';
app.dhiedralangleEditFieldLabel.Position = [1102 559 84 22];
app.dhiedralangleEditFieldLabel.Text = 'dhiedral angle ';

% Create dhiedralangleEditField
app.dhiedralangleEditField = uieditfield(app.UIFigure, 'numeric');
app.dhiedralangleEditField.Position = [1213 559 100 22];
app.dhiedralangleEditField.Value = 8;

% Create unit3_4
app.unit3_4 = uieditfield(app.UIFigure, 'text');
app.unit3_4.FontAngle = 'italic';
app.unit3_4.BackgroundColor = [0.7804 0.9098 1];
app.unit3_4.Position = [1313 559 31 22];
app.unit3_4.Value = 'deg';

% Create payloadmassEditFieldLabel
app.payloadmassEditFieldLabel = uilabel(app.UIFigure);
app.payloadmassEditFieldLabel.HorizontalAlignment = 'right';
app.payloadmassEditFieldLabel.Position = [29 567 82 22];
app.payloadmassEditFieldLabel.Text = 'payload mass ';

% Create payloadmassEditField
app.payloadmassEditField = uieditfield(app.UIFigure, 'numeric');
app.payloadmassEditField.FontSize = 14;
app.payloadmassEditField.Position = [126 567 100 22];

% Create unit2
app.unit2 = uieditfield(app.UIFigure, 'text');
app.unit2.FontSize = 14;
app.unit2.FontAngle = 'italic';
app.unit2.BackgroundColor = [0.7804 0.9098 1];
app.unit2.Position = [225 567 26 23];
app.unit2.Value = 'kg';

% Create batterymassEditField_2Label
app.batterymassEditField_2Label = uilabel(app.UIFigure);
app.batterymassEditField_2Label.HorizontalAlignment = 'right';
app.batterymassEditField_2Label.Position = [701 529 74 22];
app.batterymassEditField_2Label.Text = 'battery mass';

% Create batterymassEditField
```

```
app.batterymassEditField = uieditfield(app.UIFigure, 'numeric');
app.batterymassEditField.Position = [898 529 100 22];
app.batterymassEditField.Value = 1;

% Create unit1_3
app.unit1_3 = uieditfield(app.UIFigure, 'text');
app.unit1_3.FontAngle = 'italic';
app.unit1_3.BackgroundColor = [0.7804 0.9098 1];
app.unit1_3.Position = [997 529 26 22];
app.unit1_3.Value = 'kg';

% Create tiltangleEditFieldLabel
app.tiltangleEditFieldLabel = uilabel(app.UIFigure);
app.tiltangleEditFieldLabel.HorizontalAlignment = 'right';
app.tiltangleEditFieldLabel.Position = [1101 527 50 22];
app.tiltangleEditFieldLabel.Text = 'tilt angle';

% Create tiltangleEditField
app.tiltangleEditField = uieditfield(app.UIFigure, 'numeric');
app.tiltangleEditField.Position = [1213 528 100 22];
app.tiltangleEditField.Value = 3;

% Create unit3_3
app.unit3_3 = uieditfield(app.UIFigure, 'text');
app.unit3_3.FontAngle = 'italic';
app.unit3_3.BackgroundColor = [0.7804 0.9098 1];
app.unit3_3.Position = [1314 527 31 22];
app.unit3_3.Value = 'deg';

% Create avionicsmassEditFieldLabel
app.avionicsmassEditFieldLabel = uilabel(app.UIFigure);
app.avionicsmassEditFieldLabel.HorizontalAlignment = 'right';
app.avionicsmassEditFieldLabel.Position = [30 536 87 22];
app.avionicsmassEditFieldLabel.Text = 'avionics mass  ';

% Create avionicsmassEditField
app.avionicsmassEditField = uieditfield(app.UIFigure, 'numeric');
app.avionicsmassEditField.FontSize = 14;
app.avionicsmassEditField.Position = [126 537 100 22];

% Create unit3
app.unit3 = uieditfield(app.UIFigure, 'text');
app.unit3.FontSize = 14;
app.unit3.FontAngle = 'italic';
app.unit3.BackgroundColor = [0.7804 0.9098 1];
app.unit3.Position = [225 537 26 22];
app.unit3.Value = 'kg';

% Create avionicspowerconsumptionEditFieldLabel
app.avionicspowerconsumptionEditFieldLabel = uilabel(app.UIFigure);
app.avionicspowerconsumptionEditFieldLabel.HorizontalAlignment = 'right';
app.avionicspowerconsumptionEditFieldLabel.Position = [701 495 156 22];
app.avionicspowerconsumptionEditFieldLabel.Text = 'avionics power consumption';

% Create avionicspowerconsumptionEditField
app.avionicspowerconsumptionEditField = uieditfield(app.UIFigure, 'numeric');
app.avionicspowerconsumptionEditField.Position = [898 494 100 22];
app.avionicspowerconsumptionEditField.Value = 5;

% Create avionic_power_consumption_unit
app.avionic_power_consumption_unit = uieditfield(app.UIFigure, 'text');
app.avionic_power_consumption_unit.FontAngle = 'italic';
app.avionic_power_consumption_unit.BackgroundColor = [0.7804 0.9098 1];
app.avionic_power_consumption_unit.Position = [997 462 26 22];
app.avionic_power_consumption_unit.Value = 'W';
```

```
            % Create propulsionunitDropDownLabel
            app.propulsionunitDropDownLabel = uilabel(app.UIFigure);
            app.propulsionunitDropDownLabel.HorizontalAlignment = 'right';
            app.propulsionunitDropDownLabel.Position = [1102 493 86 22];
            app.propulsionunitDropDownLabel.Text = 'propulsion unit ';

            % Create propulsionunitDropDown
            app.propulsionunitDropDown = uidropdown(app.UIFigure);
            app.propulsionunitDropDown.Items = {'T-Motor AT2814 900KV_ Graupner CAM Folding
Prop 10X5 25X12_Zubax Myxa A2', 'MR1_T-motor AT2814 KV900 Cam-Carbon Z 9X5 23X12 test test
data', 'MR3_T-motor AT2814 KV900 Cam-Carbon Z 10X5 25X12 test test data', 'MR2_Scorpion A-5025-
220kv - Tribunus ESC - Xoar 21x8E test data', 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop
10X6-3 25X16_ Zubax Myxa A2', 'T-Motor AT2814 900KV_ Graupner Cam-Carbon Z 9X5 23X12_ Zubax
Myxa A2', 'T-Motor AT2814 900KV_Graupner CAM Folding Prop 10X6 25X15_Zubax Myxa A2', 'Scorpion
Power System LTD IM-8012-115kv_ Scorpion Power System LTD PF 28_ Scorpion Power System LTD
Legatus 12-60A FOC'};
            app.propulsionunitDropDown.DropDownOpeningFcn = createCallbackFcn(app,
@propulsionunitDropDownOpening, true);
            app.propulsionunitDropDown.ValueChangedFcn = createCallbackFcn(app,
@propulsionunitDropDownValueChanged, true);
            app.propulsionunitDropDown.Position = [1203 493 292 22];
            app.propulsionunitDropDown.Value = 'T-Motor AT2814 900KV_ Graupner CAM Folding Prop
10X5 25X12_Zubax Myxa A2';

            % Create paylodpowerconsumptionEditFieldLabel
            app.paylodpowerconsumptionEditFieldLabel = uilabel(app.UIFigure);
            app.paylodpowerconsumptionEditFieldLabel.HorizontalAlignment = 'right';
            app.paylodpowerconsumptionEditFieldLabel.Position = [701 462 148 22];
            app.paylodpowerconsumptionEditFieldLabel.Text = 'paylod power consumption';

            % Create paylodpowerconsumptionEditField
            app.paylodpowerconsumptionEditField = uieditfield(app.UIFigure, 'numeric');
            app.paylodpowerconsumptionEditField.Position = [898 462 100 22];

            % Create pay_pow_cons_unit
            app.pay_pow_cons_unit = uieditfield(app.UIFigure, 'text');
            app.pay_pow_cons_unit.FontAngle = 'italic';
            app.pay_pow_cons_unit.BackgroundColor = [0.7804 0.9098 1];
            app.pay_pow_cons_unit.Position = [997 494 26 22];
            app.pay_pow_cons_unit.Value = 'W';

            % Create calculateButton
            app.calculateButton = uibutton(app.UIFigure, 'push');
            app.calculateButton.ButtonPushedFcn = createCallbackFcn(app,
@calculateButtonPushed, true);
            app.calculateButton.BackgroundColor = [0.8588 0.7255 0.4392];
            app.calculateButton.Position = [557 362 150 30];
            app.calculateButton.Text = 'calculate';

            % Create warning
            app.warning = uieditfield(app.UIFigure, 'text');
            app.warning.Position = [707 363 150 29];

            % Create SYSTEMMASSLabel
            app.SYSTEMMASSLabel = uilabel(app.UIFigure);
            app.SYSTEMMASSLabel.FontSize = 14;
            app.SYSTEMMASSLabel.FontWeight = 'bold';
            app.SYSTEMMASSLabel.Position = [15 256 107 22];
            app.SYSTEMMASSLabel.Text = 'SYSTEM MASS';

            % Create ROTORPERFORMANCELabel
            app.ROTORPERFORMANCELabel = uilabel(app.UIFigure);
            app.ROTORPERFORMANCELabel.FontSize = 14;
            app.ROTORPERFORMANCELabel.FontWeight = 'bold';
```

```
app.ROTORPERFORMANCELabel.Position = [351 256 169 22];
app.ROTORPERFORMANCELabel.Text = 'ROTOR PERFORMANCE';

% Create POWERBUDGETLabel
app.POWERBUDGETLabel = uilabel(app.UIFigure);
app.POWERBUDGETLabel.FontSize = 14;
app.POWERBUDGETLabel.FontWeight = 'bold';
app.POWERBUDGETLabel.Position = [701 256 121 22];
app.POWERBUDGETLabel.Text = 'POWER BUDGET';

% Create HOVERPERFORMANCELabel
app.HOVERPERFORMANCELabel = uilabel(app.UIFigure);
app.HOVERPERFORMANCELabel.FontSize = 14;
app.HOVERPERFORMANCELabel.FontWeight = 'bold';
app.HOVERPERFORMANCELabel.Position = [1105 256 168 22];
app.HOVERPERFORMANCELabel.Text = 'HOVER PERFORMANCE';

% Create takeoffmassLabel
app.takeoffmassLabel = uilabel(app.UIFigure);
app.takeoffmassLabel.HorizontalAlignment = 'right';
app.takeoffmassLabel.Position = [15 187 77 22];
app.takeoffmassLabel.Text = 'take-off mass';

% Create takeoffmassEditField
app.takeoffmassEditField = uieditfield(app.UIFigure, 'numeric');
app.takeoffmassEditField.Position = [147 187 100 22];

% Create unit6
app.unit6 = uieditfield(app.UIFigure, 'text');
app.unit6.FontAngle = 'italic';
app.unit6.BackgroundColor = [0.7804 0.9098 1];
app.unit6.Position = [246 187 26 22];
app.unit6.Value = 'kg';

% Create rotorspeedEditFieldLabel
app.rotorspeedEditFieldLabel = uilabel(app.UIFigure);
app.rotorspeedEditFieldLabel.HorizontalAlignment = 'right';
app.rotorspeedEditFieldLabel.Position = [352 187 66 22];
app.rotorspeedEditFieldLabel.Text = 'rotor speed';

% Create rotorspeedEditField
app.rotorspeedEditField = uieditfield(app.UIFigure, 'numeric');
app.rotorspeedEditField.Position = [433 187 100 22];

% Create pay_pow_cons_unit_3
app.pay_pow_cons_unit_3 = uieditfield(app.UIFigure, 'text');
app.pay_pow_cons_unit_3.FontAngle = 'italic';
app.pay_pow_cons_unit_3.BackgroundColor = [0.7804 0.9098 1];
app.pay_pow_cons_unit_3.Position = [533 187 44 22];
app.pay_pow_cons_unit_3.Value = 'rpm';

% Create temperatureEditFieldLabel
app.temperatureEditFieldLabel = uilabel(app.UIFigure);
app.temperatureEditFieldLabel.HorizontalAlignment = 'right';
app.temperatureEditFieldLabel.Position = [1101 189 73 22];
app.temperatureEditFieldLabel.Text = 'temperature ';

% Create temperatureEditField
app.temperatureEditField = uieditfield(app.UIFigure, 'numeric');
app.temperatureEditField.Position = [1198 189 100 22];

% Create unit_temp
app.unit_temp = uieditfield(app.UIFigure, 'text');
app.unit_temp.FontAngle = 'italic';
app.unit_temp.BackgroundColor = [0.7804 0.9098 1];
```

```
app.unit_temp.Position = [1298 188 26 23];
app.unit_temp.Value = '°C';

% Create propulsionunitmassEditFieldLabel
app.propulsionunitmassEditFieldLabel = uilabel(app.UIFigure);
app.propulsionunitmassEditFieldLabel.HorizontalAlignment = 'right';
app.propulsionunitmassEditFieldLabel.Position = [15 157 115 22];
app.propulsionunitmassEditFieldLabel.Text = 'propulsion unit mass';

% Create propulsionunitmassEditField
app.propulsionunitmassEditField = uieditfield(app.UIFigure, 'numeric');
app.propulsionunitmassEditField.Position = [145 157 100 22];

% Create unit4
app.unit4 = uieditfield(app.UIFigure, 'text');
app.unit4.FontAngle = 'italic';
app.unit4.BackgroundColor = [0.7804 0.9098 1];
app.unit4.Position = [246 157 26 22];
app.unit4.Value = 'kg';

% Create rotortorqueEditFieldLabel
app.rotortorqueEditFieldLabel = uilabel(app.UIFigure);
app.rotortorqueEditFieldLabel.HorizontalAlignment = 'right';
app.rotortorqueEditFieldLabel.Position = [352 157 67 22];
app.rotortorqueEditFieldLabel.Text = 'rotor torque';

% Create rotortorqueEditField
app.rotortorqueEditField = uieditfield(app.UIFigure, 'numeric');
app.rotortorqueEditField.Position = [434 157 100 22];

% Create pay_pow_cons_unit_4
app.pay_pow_cons_unit_4 = uieditfield(app.UIFigure, 'text');
app.pay_pow_cons_unit_4.FontAngle = 'italic';
app.pay_pow_cons_unit_4.BackgroundColor = [0.7804 0.9098 1];
app.pay_pow_cons_unit_4.Position = [533 157 44 22];
app.pay_pow_cons_unit_4.Value = 'N m';

% Create propulsionunitpowerEditFieldLabel
app.propulsionunitpowerEditFieldLabel = uilabel(app.UIFigure);
app.propulsionunitpowerEditFieldLabel.HorizontalAlignment = 'right';
app.propulsionunitpowerEditFieldLabel.Position = [701 157 119 22];
app.propulsionunitpowerEditFieldLabel.Text = 'propulsion unit power';

% Create voltageEditFieldLabel
app.voltageEditFieldLabel = uilabel(app.UIFigure);
app.voltageEditFieldLabel.HorizontalAlignment = 'right';
app.voltageEditFieldLabel.Position = [701 187 44 22];
app.voltageEditFieldLabel.Text = 'voltage';

% Create voltageEditField
app.voltageEditField = uieditfield(app.UIFigure, 'numeric');
app.voltageEditField.Position = [834 187 100 22];

% Create propulsionunitpowerEditField
app.propulsionunitpowerEditField = uieditfield(app.UIFigure, 'numeric');
app.propulsionunitpowerEditField.Position = [835 157 100 22];

% Create unit3_5
app.unit3_5 = uieditfield(app.UIFigure, 'text');
app.unit3_5.FontAngle = 'italic';
app.unit3_5.BackgroundColor = [0.7804 0.9098 1];
app.unit3_5.Position = [934 187 26 22];
app.unit3_5.Value = 'V';

% Create pay_pow_cons_unit_2
```

```
app.pay_pow_cons_unit_2 = uieditfield(app.UIFigure, 'text');
app.pay_pow_cons_unit_2.FontAngle = 'italic';
app.pay_pow_cons_unit_2.BackgroundColor = [0.7804 0.9098 1];
app.pay_pow_cons_unit_2.Position = [934 157 26 22];
app.pay_pow_cons_unit_2.Value = 'W';

% Create epsilonEditFieldLabel
app.epsilonEditFieldLabel = uilabel(app.UIFigure);
app.epsilonEditFieldLabel.HorizontalAlignment = 'right';
app.epsilonEditFieldLabel.Position = [1103 126 43 22];
app.epsilonEditFieldLabel.Text = 'epsilon';

% Create deltaEditFieldLabel
app.deltaEditFieldLabel = uilabel(app.UIFigure);
app.deltaEditFieldLabel.HorizontalAlignment = 'right';
app.deltaEditFieldLabel.Position = [1101 157 31 22];
app.deltaEditFieldLabel.Text = 'delta';

% Create deltaEditField
app.deltaEditField = uieditfield(app.UIFigure, 'numeric');
app.deltaEditField.Position = [1198 157 100 22];

% Create epsilonEditField
app.epsilonEditField = uieditfield(app.UIFigure, 'numeric');
app.epsilonEditField.Position = [1198 126 100 22];

% Create Label_2
app.Label_2 = uilabel(app.UIFigure);
app.Label_2.HorizontalAlignment = 'right';
app.Label_2.Position = [15 127 96 22];
app.Label_2.Text = 'powerplant mass';

% Create powerplantmassEditField
app.powerplantmassEditField = uieditfield(app.UIFigure, 'numeric');
app.powerplantmassEditField.Position = [146 127 100 22];

% Create unit5
app.unit5 = uieditfield(app.UIFigure, 'text');
app.unit5.FontAngle = 'italic';
app.unit5.BackgroundColor = [0.7804 0.9098 1];
app.unit5.Position = [246 127 26 22];
app.unit5.Value = 'kg';

% Create rotorthrustEditFieldLabel
app.rotorthrustEditFieldLabel = uilabel(app.UIFigure);
app.rotorthrustEditFieldLabel.HorizontalAlignment = 'right';
app.rotorthrustEditFieldLabel.Position = [352 126 63 22];
app.rotorthrustEditFieldLabel.Text = 'rotor thrust';

% Create rotorthrustEditField
app.rotorthrustEditField = uieditfield(app.UIFigure, 'numeric');
app.rotorthrustEditField.Position = [433 127 100 22];

% Create unit1_2
app.unit1_2 = uieditfield(app.UIFigure, 'text');
app.unit1_2.FontAngle = 'italic';
app.unit1_2.BackgroundColor = [0.7804 0.9098 1];
app.unit1_2.Position = [533 126 26 22];
app.unit1_2.Value = 'N';

% Create batterypowerEditFieldLabel
app.batterypowerEditFieldLabel = uilabel(app.UIFigure);
app.batterypowerEditFieldLabel.HorizontalAlignment = 'right';
app.batterypowerEditFieldLabel.Position = [701 126 81 22];
app.batterypowerEditFieldLabel.Text = 'battery power ';
```

```
        % Create batterypowerEditField
        app.batterypowerEditField = uieditfield(app.UIFigure, 'numeric');
        app.batterypowerEditField.Position = [834 126 100 22];

        % Create pay_pow_cons_unit_5
        app.pay_pow_cons_unit_5 = uieditfield(app.UIFigure, 'text');
        app.pay_pow_cons_unit_5.FontAngle = 'italic';
        app.pay_pow_cons_unit_5.BackgroundColor = [0.7804 0.9098 1];
        app.pay_pow_cons_unit_5.Position = [934 126 26 22];
        app.pay_pow_cons_unit_5.Value = 'W';

        % Create flighttimeEditFieldLabel
        app.flighttimeEditFieldLabel = uilabel(app.UIFigure);
        app.flighttimeEditFieldLabel.HorizontalAlignment = 'right';
        app.flighttimeEditFieldLabel.Position = [1101 64 56 22];
        app.flighttimeEditFieldLabel.Text = 'flight time';

        % Create betaEditFieldLabel
        app.betaEditFieldLabel = uilabel(app.UIFigure);
        app.betaEditFieldLabel.HorizontalAlignment = 'right';
        app.betaEditFieldLabel.Position = [1101 95 28 22];
        app.betaEditFieldLabel.Text = 'beta';

        % Create betaEditField
        app.betaEditField = uieditfield(app.UIFigure, 'numeric');
        app.betaEditField.Position = [1198 95 100 22];

        % Create flighttimeEditField
        app.flighttimeEditField = uieditfield(app.UIFigure, 'numeric');
        app.flighttimeEditField.Position = [1198 64 100 22];

        % Create unit3_6
        app.unit3_6 = uieditfield(app.UIFigure, 'text');
        app.unit3_6.FontAngle = 'italic';
        app.unit3_6.BackgroundColor = [0.7804 0.9098 1];
        app.unit3_6.Position = [1298 64 41 22];
        app.unit3_6.Value = 'min';

        % Create DropDownLabel
        app.DropDownLabel = uilabel(app.UIFigure);
        app.DropDownLabel.HorizontalAlignment = 'right';
        app.DropDownLabel.Position = [772 -32 65 22];
        app.DropDownLabel.Text = 'Drop Down';

        % Create DropDown
        app.DropDown = uidropdown(app.UIFigure);
        app.DropDown.Position = [852 -32 100 22];

        % Show the figure after all components are created
        app.UIFigure.Visible = 'on';
    end
end

% App creation and deletion
methods (Access = public)

    % Construct app
    function app = app4

        % Create UIFigure and components
        createComponents(app)

        % Register the app with App Designer
        registerApp(app, app.UIFigure)
```

```
            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Bibliografia

1. E.L. de Angelis, F. Giulietti, G. Rossetti, Multirotor aircraft formation flight con-trol with collision avoidance capability, Aerosp. Sci. Technol. 77 (June 2018) 733–741, https://doi .org /10 .1016 /j .ast .2018 .04 .002.

2. K. Valavanis, G.J. Vachtsevanos, Handbook of Unmanned Aerial Vehicles, vol. 1, Springer, Netherlands, 2015, pp.308–309.

3. E.L. de Angelis, et al., Optimal autonomous multirotor motion planning in an obstructed environment, Aerosp. Sci. Technol. 87 (April 2019) 379–388, https://doi .org /10 .1016 /j .ast .2019 .03 .017.

4. E.L. de Angelis, Stability analysis of a multirotor vehicle hovering condition, Aerosp. Sci. Technol. 72 (Jan. 2018) 248–255, https://doi .org /10 .1016 /j .ast .2017.11.017.

5. Z. Zhang, J. Li, J. Wang, Sequential convex programming for nonlinear optimal control problems in UAV path planning, Aerosp. Sci. Technol. 76 (May 2018) 280–290, https://doi .org /10 .1016 /j .ast .2018 .01.040.

6. S. Harwey, A. Lucieer, Assessing the accuracy of georeferenced point clouds produced via multi–view stereopsis from Unmanned Aerial Vehicle (UAV) imagery, Remote Sens. 4(6) (2012) 1573–1599, https://doi .org /10 .3390 /rs4061573.

7. G. Cai, J. Dias, L. Seneviratne, A survey of small–scale unmanned aerial vehicles: recent advances and future development trends, Unmanned Syst. 2(2) (2014) 175–199, https://doi .org /10 .1142 /S2301385014300017.

8. M.H. Hwang, H.R. Cha, S.Y. Jung, Practical endurance estimation for minimizing energy consumption of multirotor unmanned aerial vehicles, Energies 11(9) (Aug. 2018) 1–11, https://doi .org /10 .3390 /en11092221.

9. M. Cerny, C. Breitsamter, Investigation of small–scale propellers under non–axial inflow conditions, Aerosp. Sci. Technol. 106 (Nov. 2020) 1–14, https://doi .org /10 .1016 /j .ast .2020 .106048.

10. Tyto Robotics database online https://database.tytorobotics.com/tests

11. R.W. Deters, G.K. Ananda, M.S. Selig, Reynolds number effects on the perfor-mance of small–scale propellers, in: 32nd AIAA Applied Aerodynamics Confer-ence, June 16–20, Atlanta, GA, 2014, pp.1–43.

12. S.F. Hoerner, Fluid Dynamic Drag: Practical Information on Aerodynamic Drag and Hydrodynamic Resistance, Hoerner Fluid Dynamics, 1965. Published by the Author, Bakersfield, CA, Ch. 2.

13. Z.J. Chen, A. Stol, P.J. Richards, Preliminary design of multirotor UAVs with tilted–rotors for improved disturbance rejection capability, Aerosp. Sci. Tech-nol. 92 (2019) 635–643, https://doi .org /10 .1016 /j .ast .2019 .06 .038.

14. RCTimer, Commercial/Industrial Propellers, Retrieved from https://www.rctimer.com /commercial -industrial -propellers -c0420, 2021.

15. T–Motor, Carbon Fiber and Polymer Propellers, Retrieved from https://store -en .tmotor.com /category.php ?id =4, 2021.

16. G. Avanzini, E.L. de Angelis, F. Giulietti, Optimal performance and sizing of a battery–powered aircraft, Aerosp. Sci. Technol. 59 (2016) 132–144, https://doi .org /10 .1016 /j .ast .2016 .10 .015.

17. 17. Performance analysis and optimal sizing of electric multirotors Emanuele L. de Angelis, Fa- brizio Giulietti, Gianluca Rossetti, Gabriele Bellani University of Bologna, Forlı, 47121, Italy