UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

Implementazione firmware di un microcontrollore per il controllo del volo di un drone a flusso convogliato

Firmware programming of a microcontroller for the flight control of a duct fan drone

Relatore:
Prof. Andrea Bonci

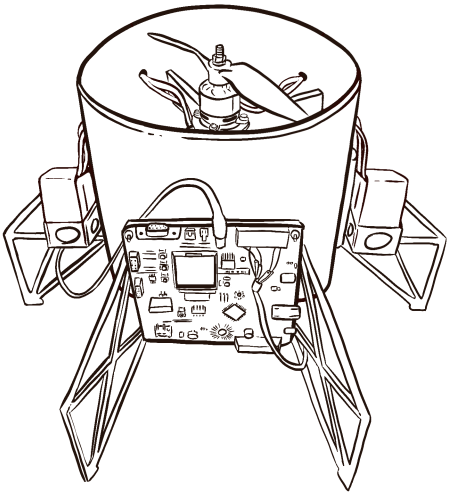
Tesi di Laurea di:
Daniele Di Muzio

Anno Accademico 2019/2020

Abstract

di Daniele Di Muzio

L'elaborato tratta lo sviluppo del software di controllo di un drone a flusso convogliato. Partendo dalla descrizione dei componenti hardware e dall'analisi del software, si giunge alla versione finale del codice implementato per il controllo del drone, riportando dettagliatamente i test effettuati per verificarne il corretto funzionamento.



Indice

Capitolo 1

Introduzione

Lo scopo della tesi è stato quello di sviluppare il software per il controllo del volo di un drone a flusso convogliato e successivamente effettuare delle prove di test per verificare il corretto funzionamento del software stesso. Con l'utilizzo di droni che negli ultimi anni sta crescendo sempre di più è utile studiare un design che permetta di rendere sempre più sicuro il volo di questi velivoli senza avere la necessità di preoccuparsi dell'ambiente circostante. I droni a flusso convogliato riescono a soddisfare questa necessità in quanto le eliche sono poste all'interno della struttura che, oltre a convogliare correttamente l'aria per generare un flusso sufficiente a far si che il drone si sollevi dal suolo, funge essa stessa da protezione per le eliche verso l'ambiente circostante.
Lo sviluppo del software è stato realizzato andando a studiare ed incorporare nella maniera corretta parti di codice dei singoli componenti e sensori presenti nel drone.
La parte di test è stata suddivisa in due blocchi, un primo blocco prevede il test dei singoli componenti e un secondo blocco prevede il test di funzionamento complessivo del drone.
La tesi è strutturata come segue:

- nel secondo capitolo, *Struttura del drone*, viene descritta la struttura del drone;
- nel terzo capitolo, *Architettura hardware*, vengono descritte tutte le componenti hardware e i sensori utilizzati per lo sviluppo del drone;
- nel quarto capitolo, *Architettura software*, viene descritto lo sviluppo del software di controllo del drone;

- nel quinto capitolo, *Prove sperimentali*, vengono riportate le prove di test effettuate per la verifica del software di controllo;
- nel sesto capitolo, *Conclusioni*, vengono riportati i risultati ottenuti e i possibili sviluppi futuri;

1.1 Equazioni del moto

In questa tesi sono illustrate le modalità di sviluppo del software di controllo di un drone a flusso convogliato ma verrà fatta una breve trattazione sullo studio delle equazioni del moto che caratterizzano un ducted fan a doppia elica con un solo livello di flap. Prima di descrivere brevemente le equazioni, si elencano di seguito tutte le costanti e le variabili utilizzate per esplicitare le formule.

Tabella 1.1: Variabili

parametro	notazione	u.m.(SI)
u	velocità del baricentro lungo l'asse x espressa rispetto alla terna solidale al corpo	$\frac{m}{s}$
v	velocità del baricentro lungo l'asse y espressa rispetto alla terna solidale al corpo	$\frac{m}{s}$
w	velocità del baricentro lungo l'asse z espressa rispetto alla terna solidale al corpo	$\frac{m}{s}$
$\dot{u}$	accelerazione del baricentro lungo l'asse x espressa rispetto alla terna solidale al corpo	$\frac{m}{s^2}$
$\dot{v}$	accelerazione del baricentro lungo l'asse y espressa rispetto alla terna solidale al corpo	$\frac{m}{s^2}$
$\dot{w}$	accelerazione del baricentro lungo l'asse z espressa rispetto alla terna solidale al corpo	$\frac{m}{s^2}$
w_x	velocità angolare lungo l'asse x rispetto alla terna del corpo	$\frac{rad}{s}$
w_y	velocità angolare lungo l'asse y rispetto alla terna del corpo	$\frac{rad}{s}$
w_z	velocità angolare lungo l'asse z	$\frac{rad}{s}$

Tabella 1.1: continua

Tabella 1.1: continua

parametro	notazione	u.m.(SI)
	rispetto alla terna del corpo	
$\dot{w}_x$	accelerazione angolare lungo l'asse x	$\frac{rad}{s^2}$
	rispetto alla terna del corpo	
$\dot{w}_y$	accelerazione angolare lungo l'asse y	$\frac{rad}{s^2}$
	rispetto alla terna del corpo	
$\dot{w}_z$	accelerazione angolare lungo l'asse z	$\frac{rad}{s^2}$
	rispetto alla terna del corpo	
$\dot{\Phi}$	velocità angolare lungo l'asse x	$\frac{rad}{s}$
	rispetto al sistema non inerziale	
$\dot{\Theta}$	velocità angolare lungo l'asse y	$\frac{rad}{s}$
	rispetto al sistema non inerziale	
$\dot{\Psi}$	velocità angolare lungo l'asse z	$\frac{rad}{s}$
	rispetto al sistema non inerziale	
F_x	Forza lungo l'asse x	N
F_y	Forza lungo l'asse y	N
F_z	Forza lungo l'asse z	N
τ_x	Componente di coppia lungo l'asse x	N
τ_y	Componente di coppia lungo l'asse y	N
τ_z	Componente di coppia lungo l'asse z	N
$\dot{x}_G$	velocità lungo l'asse x espressa rispetto	$\frac{m}{s}$
	al sistema di riferimento inerziale	
$\dot{y}_G$	velocità lungo l'asse y espressa rispetto	$\frac{m}{s}$
	al sistema di riferimento inerziale	
$\dot{z}_G$	velocità lungo l'asse z espressa rispetto	$\frac{m}{s}$
	al sistema di riferimento inerziale	
T_u	Thrust motore superiore (upper)	N
T_l	Thrust motore inferiore (lower)	N

Per descrivere le equazioni del modello vengono usate le seguenti costanti:

Grandezza	Valore	Notazione	u.m.(SI)
m	1.854	massa del velivolo	kg
g	9.81	accelerazione di gravità	$\frac{m}{s^2}$
I_{xx}	$1.8 \cdot 10^{-2}$	Momento d'inerzia rispetto all'asse x	$kg \cdot m^2$
I_{yy}	$1.8 \cdot 10^{-2}$	Momento d'inerzia rispetto all'asse y	$kg \cdot m^2$
I_{zz}	$1.8 \cdot 10^{-2}$	Momento d'inerzia rispetto all'asse z	$kg \cdot m^2$
k_T	$7.6 \cdot 10^{-5}$	Costante di spinta	$kg \cdot m$
k_N	$9.07 \cdot 10^{-7}$	Costante di coppia di reazione	$kg \cdot m^2$

Infine, per semplificare la lettura delle equazioni del velivolo, è stata scelta un formato di scrittura compatta per le formule trigonometriche, come viene riportato nella tabella seguente:

Grandezza	Forma compatta
$sin\,(\Phi)$	s_Φ
$cos\,(\Phi)$	c_Φ
$sin\,(\Theta)$	s_Θ
$cos\,(\Theta)$	c_Θ
$tan\,(\Theta)$	t_Θ
$sin\,(\Psi)$	s_Ψ
$cos\,(\Psi)$	c_Ψ

Una volta introdotte tutte le grandezze utilizzate si procede con lo studio delle equazioni del moto.

1.1.1 Modello generale

Si parte dallo studio generale del modello matematico di un generico velivolo sul quale però non si considerano le forze e i momenti a cui quest'ultimo è soggetto in quanto dipendono dalla struttura specifica che si considera.
La dinamica traslazionale è descritta dalle seguenti equazioni [1]:

$$\dot{u} = \omega_z v - \omega_y w + \frac{F_x}{m} \tag{1.1}$$

$$\dot{v} = -\omega_z u + \omega_x w + \frac{F_y}{m} \tag{1.2}$$

$$\dot{w} = \omega_y u - \omega_x v + \frac{F_z}{m} \tag{1.3}$$

La dinamica rotazionale, invece, è descritta dalle seguenti equazioni:

$$\dot{\omega_x} = \frac{1}{I_{xx}} \left[\omega_z \omega_y + \tau_x\right] \tag{1.4}$$

$$\dot{\omega_y} = \frac{1}{I_{yy}} \left[\omega_x \omega_z \left(I_{zz} - I_{xx}\right) + \tau_y\right] \tag{1.5}$$

$$\dot{\omega_z} = \frac{1}{I_{zz}} \left[\omega_x \omega_y \left(I_{xx} - I_{yy}\right) + \tau_z\right] \tag{1.6}$$

Le equazioni cinematiche, scegliendo le orientazioni RPY[1] per l'orientamento, sono:

$$\dot{\phi} = \omega_x + \omega_y s_\phi t_\theta + \omega_z c_\phi t_\theta \tag{1.7}$$

$$\dot{\theta} = \omega_y c_\phi - \omega_z s_\phi \tag{1.8}$$

$$\dot{\psi} = \omega_y \frac{s_\phi}{c_\theta} + \omega_z \frac{c_\phi}{c_\theta} \tag{1.9}$$

Le equazioni enunciate fino ad ora sono scritte rispetto al sistema di riferimento del corpo; utilizzando le equazioni di navigazione si riportano le velocità del baricentro espresse rispetto a questo sistema di riferimento $(\dot{u}, \dot{v}, \dot{w})^T$ nelle corrispondenti velocità espresse rispetto al sistema di riferimento inerziale $(\dot{x_G}, \dot{y_G}, \dot{z_G})^T$:

$$\dot{x_G} = w \left(s_\phi s_\psi + c_\phi c_\psi s_\theta\right) - v \left(c_\phi s_\psi - c_\psi s_\phi s_\theta\right) + u c_\psi c_\theta \tag{1.10}$$

$$\dot{y_G} = v \left(c_\phi c_\psi + s_\phi s_\psi s_\theta\right) - w \left(c_\psi s_\phi - c_\phi s_\psi s_\theta\right) + u c_\theta s_\psi \tag{1.11}$$

[1] Roll, pitch e yaw

$$\dot{z_G} = wc_\phi c_\theta - us_\theta + vc_\theta s_\phi \tag{1.12}$$

1.1.2 Ducted fan a doppia elica

Dopo aver analizzato il modello matematico del microelicottero coassiale, che non sarà trattato in questo elaborato, ci si può ricondurre alle equazioni del ducted fan a doppia elica con un livello di flap che comanda gli angoli di *pitch* e *roll*. La differenza sostanziale con il microelicottero coassiale è che i piani di rotazione delle due eliche rimangono sempre paralleli tra di loro, mentre nel microelicottero coassiale i movimenti di rotazione e di traslazione vengono gestiti con l'inclinazione dei piani di rotazione delle eliche. Questo fa si che, nel ducted fan a doppia elica, il vettore di spinta è sempre diretto lungo l'asse z che risulta essere perpendicolare al piano di rotazione delle eliche. Inoltre si può ipotizzare che le distanze tra i rotori e il baricentro sono le stesse in quanto i due rotori sono vicini tra di loro. Le equazioni che si ottengono sono:

$$F_x = -mgs_\theta \tag{1.13}$$

$$F_y = mgc_\theta s_\phi \tag{1.14}$$

$$F_z = -\left(T_u + T_l\right) + mgc_\theta c_\phi = -T + mgc_\theta\, c_\phi \tag{1.15}$$

$$\tau_x = 0 \tag{1.16}$$

$$\tau_y = 0 \tag{1.17}$$

$$\tau_z = N_u - N_l = \frac{k_N}{k_T}\left(T_u - T_l\right) \tag{1.18}$$

Queste equazioni non tengono conto dei flap che gestiscono gli angoli di *pitch* e di *roll*. Per introdurli si modificano le equazioni (1.16) e (1.17), ovvero le equazioni di coppia, come segue:

$$\tau_x = -k_{a1} bT \tag{1.19}$$

$$\tau_y = k_{a1} aT \tag{1.20}$$

I momenti di rollio e di beccheggio sono dati dalle equazioni τ_x e τ_y mentre il momento di imbardata è descritto dall'equazione τ_z ed è determinato dalla differenza delle spinte dei due motori, che successivamente verrà chiamata $c = T_u - T_l$. Riassumendo le equazioni che descrivono la dinamica e la cinematica del ducted fan a doppia elica si ha che:

$$\dot{u} = \omega_z v - \omega_y w$$

$$\dot{v} = -\omega_z u + \omega_x w$$

$$\dot{w} = \omega_y u - \omega_x v - \frac{T}{m} + g c_\theta c_\phi$$

$$\dot{\omega_x} = \frac{1}{I_{xx}} \left[\omega_z \omega_y - k_{a1} b T\right]$$

$$\dot{\omega_y} = \frac{1}{I_{yy}} \left[\omega_x \omega_z \left(I_{zz} - I_{xx}\right) + k_{a1} a T\right]$$

$$\dot{\omega_z} = \frac{1}{I_{zz}} \left[\omega_x \omega_y \left(I_{xx} - I_{yy}\right) + \frac{k_N}{k_T} c\right]$$

$$\dot{\phi} = \omega_x + \omega_y s_\phi t_\theta + \omega_z c_\phi t_\theta$$

$$\dot{\theta} = \omega_y c_\phi - \omega_z s_\phi$$

$$\dot{\psi} = \omega_y \frac{s_\phi}{c_\theta} + \omega_z \frac{c_\phi}{c_\theta}$$

$$\dot{x_G} = w \left(s_\phi s_\psi + c_\phi c_\psi s_\theta\right) - v \left(c_\phi s_\psi - c_\psi s_\phi s_\theta\right) + u c_\psi c_\theta$$

$$\dot{y_G} = v \left(c_\phi c_\psi + s_\phi s_\psi s_\theta\right) - w \left(c_\psi s_\phi - c_\phi s_\psi s_\theta\right) + u c_\theta s_\psi$$

$$\dot{z_G} = w c_\phi c_\theta - u s_\theta + v c_\theta s_\phi$$

Capitolo 2

Struttura del drone

Il ducted fan è caratterizzato da una struttura cilindrica all'interno della quale sono presenti tutte le componenti necessarie per il moto del velivolo. Questa configurazione permette una maggiore sicurezza nel volo, poiché tutte le componenti in movimento del drone sono protette dall'ambiente circostante dalla struttura stessa. Nella configurazione che è stata presa in esame per questa tesi si va a studiare un ducted fan a doppia elica con un singolo livello di flap situato nella parte bassa della struttura. Quest'ultima è stata progettata e successivamente stampata in PVC mediante l'utilizzo di una stampante 3D. Una volta ottenute le singole componenti è stato possibile assemblarle tra di loro per ottenere la struttura finale.

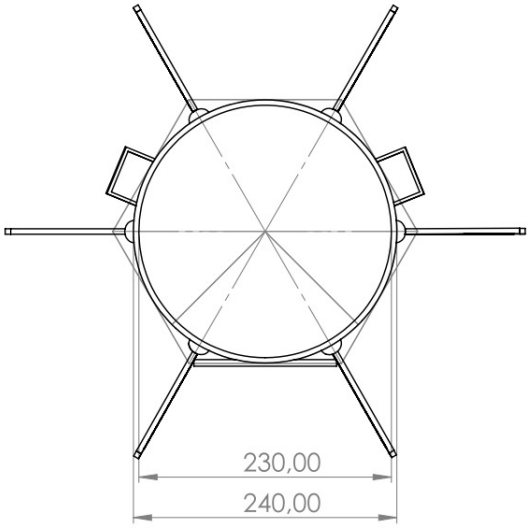


Figura 2.1: Vista superiore del ducted fan

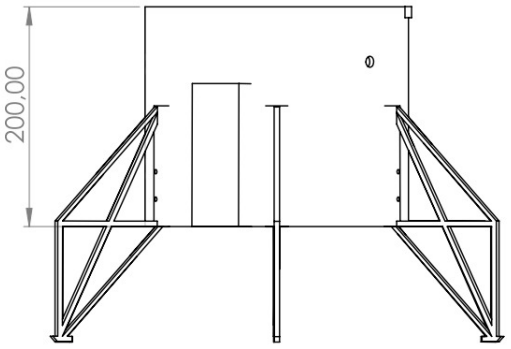


FIGURA 2.2: Vista frontale del ducted fan

Prima di descrivere la parte strutturale del drone è necessario verificare le posizioni del centro di massa e degli assi dei momenti di inerzia, fondamentali per lo studio delle equazioni riportate nel capitolo precedente.

Per calcolare il centro di massa della struttura è sufficiente utilizzare l'equazione generale, utilizzando la media delle masse pesate su ogni singola coordinata x, y, z ricavando così il punto di coordinate corrispondente al centro di massa. Con le disposizioni descritte nell'appendice A il centro di massa è stato individuato sull'asse di rotazione verticale del drone al centro della struttura.

Per il calcolo degli assi di inerzia e i relativi momenti, occorre traslare la matrice d'inerzia di ogni oggetto nel centro di massa della struttura, utilizzando il teorema di Huygens-Steiner generale per 3 assi, così da poter calcolare gli assi principali d'inerzia risolvendo le equazioni che riportano la matrice diagonale d'inerzia per le tre generiche rotazioni di *roll*, *pitch* e *yaw*.

Si prosegue con lo studio della struttura del drone, trovando nella parte interna superiore del ducted fan i due motori. Questi sono ancorati ad una struttura a croce utilizzata sia come base per i motori stessi sia come rinforzo alle sollecitazioni che può subire il drone. Realizzata e stampata anch'essa in PVC, fornisce una maggior rigidità alla parte superiore della struttura che altrimenti potrebbe essere alterata dai flussi di aria generati dalle eliche che potrebbero portare la struttura a collassare su se stessa.

Nella parte inferiore del ducted fan troviamo i due flap, uno per la gestione dell'angolo di *pitch* e un altro per l'angolo di *roll*. Entrambi i flap sono collegati ad un'estremità al servomotore che ne gestisce il movimento, mentre nell'altra sono inseriti in un apposito foro presente nella struttura che lo mantiene parallelo al terreno. Per ottenere un posizionamento corretto dei due flap, a 90° tra di loro, è stata utilizzata un'apposita placca a scorrimento, pensata e realizzata su misura per il ducted fan, alla quale sono agganciati i servomotori. La placca permette al blocco

servomotore-flap di essere spostato di qualche millimetro per ottenere una posizione ottimale che, una volta trovata, si può fissare con i bulloni presenti nella placca che garantiranno il mantenimento della corretta posizione.

All'esterno della struttura si trovano le varie componenti hardware, che verranno descritte nel capitolo successivo, e sei piedi che forniscono un appoggio stabile al terreno per l'intero drone. Sono distanziati in egual misura l'uno dall'altro per una corretta distribuzione dei pesi e sono agganciati alla struttura mediante due bulloni opportunamente posizionati con il dado all'esterno e la testa all'interno per evitare che il flusso d'aria generato nel cilindro potesse in qualche modo allentare i dadi e causare problemi di rottura o indebolimento nel tempo. I piedi sono stati anch'essi realizzati su misura e successivamente stampati in PVC per poi essere assemblati come sopra descritto.

Qui sotto vengono riportate le misure in *mm* utilizzate per la realizzazione degli appoggi.

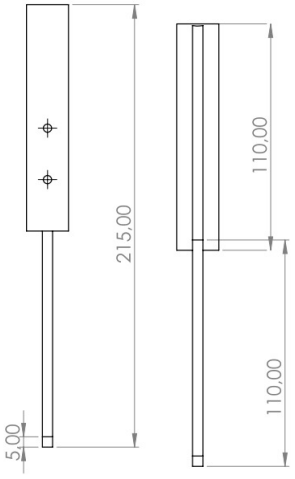


Figura 2.3: Vista frontale

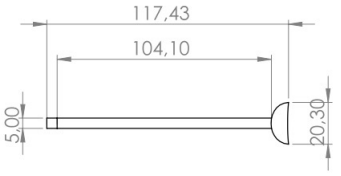


FIGURA 2.4: Vista superiore

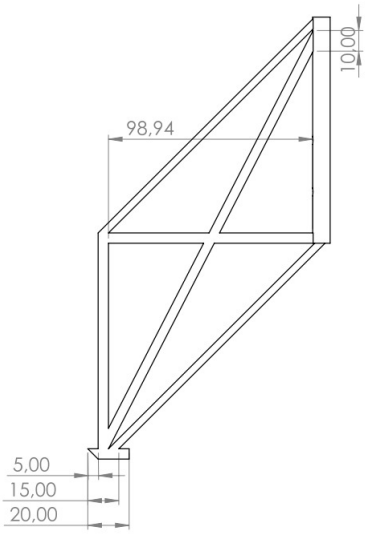


FIGURA 2.5: Vista laterale

Capitolo 3

Architettura hardware

Di seguito verranno descritte tutte le componenti hardware utilizzate per la realizzazione del progetto. Per la gestione del software e per l'acquisizione dei dati dai sensori necessari per il corretto funzionamento dei motori e dei flap del drone è stata utilizzata la scheda Renesas Demonstration Kit YRDKRX63N. Prima di essere montato a bordo della struttura, ogni componente è stato testato singolarmente, le prove verranno descritte nel capitolo 5 dell'elaborato.

3.1 Renesas Demonstration Kit YRDKRX63N

La Renesas Demonstration Kit YRDKRX63N è una scheda che si occupa della gestione del software implementato sul microcontrollore RX63N. La sua programmazione avviene mediante l'ambiente di sviluppo e^2 *studio*, tramite il quale si eseguono anche codifica e debugging del codice da implementare.

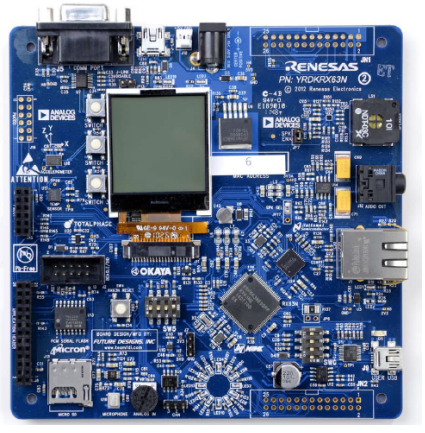


FIGURA 3.1: Renesas Demonstration Kit YRDKRX63N

3.2 Motori ed ESC

Per questo progetto sono stati utilizzati dei motori brushless Turnigy Aerodrive SK3-3536 da 1400kv.[1] Di seguito sono riportate le principali caratteristiche tecniche:

- Tensione: 11.1V~16.8V;
- Velocità angolare massima $\approx 2345.72 \frac{rad}{s}$;
- Potenza massima: 590W;
- Corrente massima: 40A;
- Peso: circa 105gr.

FIGURA 3.2: Turnigy Aerodrive SK3-3536

Per il controllo dei motori sono stati utilizzati degli ESC[2], dispositivi che permettono di gestire e regolare le velocità di rotazione andando a variare la percentuale di segnale PWM inviata ai motori.

FIGURA 3.3: Turnigy Plush-40A

[1] Con kv si indica il rapporto RPM/V
[2] Electronic Speed Control

Per il corretto funzionamento dei motori è necessaria una procedura di armamento che consiste nell'invio di un segnale PWM con un duty cycle di ampiezza del 4.75%. L'esito di queste operazioni può essere stabilito grazie a dei segnali acustici (*beep*) emessi dall'ESC. Di seguito sono riportati quelli più comuni:

- Un *beep* ogni due secondi indica l'attesa di un segnale PWM idoneo per l'armamento iniziale dei motori;
- Tre *beep* consecutivi seguiti da un *beep* prolungato indicano l'esito positivo delle operazioni di armamento;
- Una ripetizione rapida di *beep* sta ad indicare il mancato armamento dei motori dovuto ad un segnale in ingresso errato.

Una volta eseguita la procedura di armamento dei motori è necessario attendere qualche secondo per evitare che questi si disarmino.
Dopo varie prove di test si è stabilito come range di lavoro per il motore un segnale PWM che va da un duty cycle minimo del 6% ad un massimo del 10% quindi, per variare la velocità, si deve trovare il valore di duty cycle necessario per fornire la spinta adatta, questa procedura verrà descritta nel quarto capitolo, *Architettura software*. Valori superiori al 10% di duty cycle causano il disarmamento dei motori.

3.3 Eliche

Una delle scelte più importanti da fare per la corretta realizzazione del progetto è sicuramente quella delle eliche da utilizzare. Sono, abbinate al motore, la fonte primaria di erogazione della spinta necessaria per far si che il drone riesca a sollevarsi da terra. É stato quindi necessario uno studio per la corretta selezione in base a due parametri fondamentali che caratterizzano tutte le tipologie di eliche, il diametro e il passo. Con il primo si indica il diametro della circonferenza che l'elica produce quando è in rotazione, con il secondo si indica la distanza che viene percorsa dall'elica dopo un giro completo. É molto importante dimensionare nella maniera corretta questi due parametri in quanto valori troppo bassi potrebbero portare a difficoltà nel far sollevare il drone da terra, al contrario, valori troppo alti, potrebbero non sfruttare a pieno le capacità dell'elica stessa. Solitamente queste due grandezze vengono espresse in pollici[3] e servono per indicare il modello di elica utilizzata. Per questo progetto sono state scelte le GEMFAN GF 9060 9"x6".

Figura 3.4: Eliche GEMFAN GF 9060

3.4 Batteria e Power Safe module

Per ottimizzare la posizione e la corretta distribuzione dei pesi sono state utilizzate due batterie poste in parallelo tra di loro, per il posizionamento si faccia riferimento all'Appendice A. Di seguito vengono elencate le principali caratteristiche di ciascuna batteria:

- Tensione: 14.8V;

[3] *inch* in inglese, vengono indicati mediate il simbolo "

- Capacità: 1300mAh;
- Capacità di scarica[4]: 75C/150C;
- Numero di celle: 4.

FIGURA 3.5: Batteria Tattu

Essendo le due batterie poste in parallelo è necessario utilizzare un power safe module che permetta il collegamento nella maniera più sicura. Questo, inoltre, fa si che le batterie si scarichino in egual misura evitando problemi di erogazione della corrente dovuti alla diversa quantità di carica delle batterie stesse. Per evitare il danneggiamento della batteria è consigliabile non far scendere mai la tensione della batteria al di sotto dei 12V, quindi, considerando le 4 celle a disposizione, è consigliabile mantenere il valore di ogni cella sempre al di sopra sopra dei 3V.

FIGURA 3.6: MRN-electronic Power safe module

3.5 Convertitore di tensione

Per questo progetto sono stati utilizzati diversi dispositivi ognuno dei quali necessita di una tensione di alimentazione differente. É stato quindi necessario ricorrere ad un convertitore di tensione per distribuire nella maniera corretta le tensioni di alimentazione ai vari dispositivi. Il convertitore di tensione riceve in ingresso la tensione dalle batterie ed eroga in uscita quattro diversi valori di tensione a seconda dei pin che si sceglie di utilizzare. La prima soluzione che si

[4]Il primo valore indica la capacità di scarica costante, mentre il secondo è quella di picco

può adottare è quella di fornire in uscita la stessa tensione che si ha in ingresso, in questo caso il dispositivo funge solamente da ponte per il passaggio della tensione. Un'altra soluzione è quella di utilizzare degli appositi pin sul dispositivo dai quali può essere erogata una tensione di 3.3V o di 5V, selezionabile mediante un apposito switch; queste due tensioni in uscita non possono essere erogate contemporaneamente in quanto la selezione è vincolata alla posizione dello switch. Infine ci sono dei pin che erogano in uscita una tensione variabile mediante un dimmer che permette di selezionare valori intermedi di tensione fino ad un massimo pari alla tensione di ingresso. Questa soluzione è stata utilizzata per alimentare i servomotori con una tensione di 6.5V.

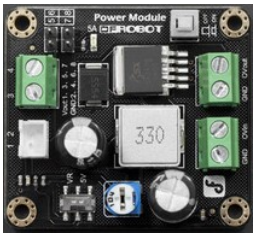


Figura 3.7: DFRobot Power module

3.6 Servomotori

Sono stati utilizzati dei servomotori DS3115 Digital, i quali permettono il movimento dei flap per la gestione degli angoli di *pitch* e di *roll* cosicché il drone rimanga stabile in aria a seguito di sollecitazioni che potrebbero portarlo in uno stato di movimento non controllato. Le specifiche di questi servomotori indicano come range di funzionamento tensioni che vanno da 5V a 6.8V, per evitare complicazioni, è stata scelta una tensione di utilizzo di 6.5V, la quale permette comunque una risposta in tempi brevi da parte del servomotore. Tensioni al di sotto dei 6V possono portare ad un aumento dei tempi di risposta che potrebbero, quindi, procurare problemi nella gestione del drone.

I servomotori sono alimentati da un segnale PWM che fa variare la posizione dei flap mediante opportune conversioni effettuate nel codice. Il range completo di movimento, che va da 0° a 180°, è stato limitato per problemi di sicurezza che potevano compromettere la struttura, in quanto, essendo i due flap sovrapposti, un movimento troppo ampio da parte di uno dei due avrebbe potuto provocare una possibile rottura della struttura. Il range di movimento è stato quindi fissato, tramite codice, tra 60° e 120° con la posizione di partenza impostata a 90° rispetto

al terreno, risultando così perpendicolare ad esso. Per una gestione ottimale degli angoli di *pitch* e di *roll* i servomotori devono essere disposti a 90° tra di loro e fissati alla struttura del drone mediante apposite placche a scorrimento per ottenere un posizionamento ottimale.

FIGURA 3.8: Servomotore DS3115 Digital

3.7 Sensore di altezza

Per il rilevamento della quota del drone è stato utilizzato il sensore di distanza VL53L1X STMicroelectronics che utilizza la tecnologia *Time of Flight* (ToF) per l'acquisizione dei dati. In questa maniera è possibile effettuare misure indipendentemente dalla riflettenza della superficie interessata, a differenza di tecnologie precedenti, come ad esempio le misurazioni ad infrarossi, le quali si limitavano a misurare l'intensità del segnale riflesso e quindi potevano dare misure alterate dalla superficie stessa. La tecnologia ToF misura la distanza sensore-oggetto calcolando il tempo che l'impulso luminoso, generato da un emettitore presente sul sensore, impiega per raggiungere l'oggetto e tornare indietro. In questo modo è possibile ricavare la distanza utilizzando la formula $d = \frac{c\Delta t}{2}$, dove c è la velocità della luce e Δt è il tempo trascorso da quando l'impulso viene inviato a quando viene ricevuto.

É possibile selezionare quattro modalità di lettura della distanza che andranno ad influire sulla precisione delle misure. Ogni modalità è più adatta ad un certo range di misure ed è caratterizzata da due parametri, il *timing budget* e l'*inter measurement*. Il primo sta ad indicare il tempo che il sensore impiega per effettuare la misura, mentre il secondo il tempo che intercorre tra una misura e la successiva. Di seguito sono riportati i range di misura più adatti alle varie distanze con le relative tempistiche:

Modalità	Range minimo [mm]	Range massimo [mm]	Tempo di misura [ms]	Timing budget [ms]
Short	0	1300	50	20
Medium	1301	1800	110	50
Long	1801	3400	110	50
LongLong	3401	4500	290	140

Tabella 3.1: Range di valori sensore di altezza

Il sensore per il rilevamento dell'altezza è posizionato sulla base del drone in parallelo rispetto al terreno e misura solamente la distanza perpendicolare tra esso e il primo oggetto che incontra. Principalmente i risultati delle misure servono per incrementare o meno la potenza da fornire ai motori per raggiungere una determinata quota di altezza, però, se i valori degli angoli di *pitch* e di *roll* sono diversi da zero, il drone sarà inclinato rispetto al terreno, di conseguenza la misura che fornirà il sensore, ovvero quella perpendicolare ad esso, non sarà quella dell'effettiva distanza tra drone e terreno. Dovrà, quindi, essere manipolata mediante il codice per ottenere il risultato corretto ma per il momento questa funzionalità non è ancora stata implementata, di conseguenza, le correzioni sulla spinta del motore che vengono effettuate non tengono conto degli angoli di inclinazione a cui è soggetto il drone. Per quote di altezza non troppo elevate questa differenza di misure può essere considerata poco rilevante ai fini generali del progetto.

Figura 3.9: sensore VL53L1X

3.8 IMU

L'IMU (Inertial Measurements Unit) è una scheda elettronica capace di misurare specifiche caratteristiche ambientali o di movimento. É un componente fondamentale per i sistemi di guida inerziali e viene spesso utilizzata in velivoli come droni per il controllo degli angoli di *roll*, *pitch*, e *yaw*.

In questo progetto è stata utilizzata una IMU Drotek 10DoF costituita dalle seguenti componenti:

- accelerometro: MPU-6050 di TDK InvenSense;
- giroscopio: MPU-6050 di TDK InvenSense;
- magnetometro: HMC5983 di Honeywell;

FIGURA 3.10: IMU Drotek 10DoF

3.8.1 Accelerometro e giroscopio

Nel circuito integrato presente all'interno del sensore MPU-6050 troviamo un accelerometro e un giroscopio entrambi a tre assi, quindi fornirà un'uscita a sei valori; l'accelerometro misura l'accelerazione del corpo lungo le tre direzioni, il giroscopio misura le accelerazioni angolari del corpo introno ai suoi tre assi.

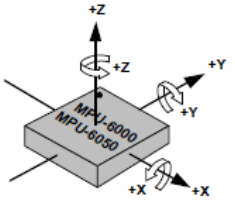


FIGURA 3.11: Orientamento degli assi di rotazione e direzione

Come già detto, l'accelerometro è un dispositivo in grado di misurare l'accelerazione andando a ricavare la forza rilevata rispetto ad una massa. Esistono diverse tipologie di accelerometri ma in questo progetto è stato utilizzato uno sviluppato con la tecnologia MEMS[5], una tecnologia capacitiva che sfrutta la variazione di capacità elettrica come misura dell'accelerazione. Nello specifico si avrà una massa formata da un'armatura mobile mentre un'altra è realizzata sul dispositivo ed è fissata ad esso. La prima massa viene sospesa su una membrana elastica che

[5]Micro Electro-Mechanical Systems

tramite un apposito circuito rileva la capacità del condensatore e genera un segnale elettrico proporzionale alla posizione della massa. Quando si verifica un'accelerazione lungo uno dei tre assi la membrana elastica subirà uno spostamento che determina una variazione di capacità nel condensatore alla quale corrisponde un preciso valore di accelerazione. Il sensore MPU-6050 utilizza masse separate per ognuno dei tre assi ottenendo così valori più precisi delle diverse accelerazioni.
Il giroscopio invece va a misurare l'accelerazione angolare lungo i tre assi di rotazione. Anche questo, come l'accelerometro, è stato realizzato utilizzando la tecnologia MEMS, in particolare utilizza l'effetto Coriolis per effettuare le proprio misure. Anche qui è presente una massa che si muove costantemente in una direzione con una determinata velocità; quando è applicata una forza esterna ci sarà uno spostamento perpendicolare della massa che causerà una variazione di capacità elettrica che verrà misurata e tramutata in velocità angolare rispetto ad uno dei tre assi.

3.8.2 Magnetometro

Il magnetometro è uno strumento utilizzato per la misurazione dell'intensità di un campo magnetico, in particolar modo quello terrestre. É formato da sensori magnetoresistivi che rilevano il cambiamento del flusso di materiali ferromagnetici nei quali avviene un cambiamento di resistenza quando viene applicato un campo magnetico perpendicolarmente al flusso di corrente in una striscia di materiale ferroso.
In questo progetto è stato utilizzato un magnetometro HMC5983 della Honeywell, un magnetometro con circuito integrato a tre assi con compensazione della temperatura.
Per il corretto funzionamento del magnetometro questo deve esse calibrato prima di ogni utilizzo andando ad eliminare tutti i contributi di campo magnetico che vadano ad influire su quello terrestre che si vuole misurare. Poiché questi contributi sono costanti e indipendenti dall'orientamento in ogni punto dello spazio, possono essere individuati e poi sottratti. Per individuare il contributo su ogni asse è necessario allineare il dispositivo in modo tale che gli assi del sensore corrispondano a quelli del campo magnetico terrestre, tuttavia in fase di calibrazione non sempre sarà nota la direzione del campo magnetico terrestre, di conseguenza è necessario registrare dati per un certo periodo di tempo. I valori che si avrebbero in corrispondenza dei tre assi possono essere individuati andando ad identificare i valori di picco rilevati. Per avere un'acquisizione ottimale dei dati di consiglia di effettuare questa procedura per una decina di secondi ruotando il sensore in modo tale che assuma tutte le posizioni rispetto ai tre assi di

rotazione. Essendo la calibrazione un'operazione necessaria prima di ogni utilizzo che però può essere scomodo effettuare una volta che il sensore è montato a bordo del drone, si è optato per il salvataggio dei valori necessari alla calibrazione in apposite costanti che vengono settate tramite codice durante l'inizializzazione dei vari sensori.

Capitolo 4

Architettura software

Per lo sviluppo completo del software per il controllo del ducted fan è stato necessario implementare nella maniera corretta e più funzionale i codici per la gestione dei vari componenti hardware. Come prima cosa è stato quindi necessario uno studio delle singole parti di codice per poter effettuare in un secondo momento il *merge* del codice completo.

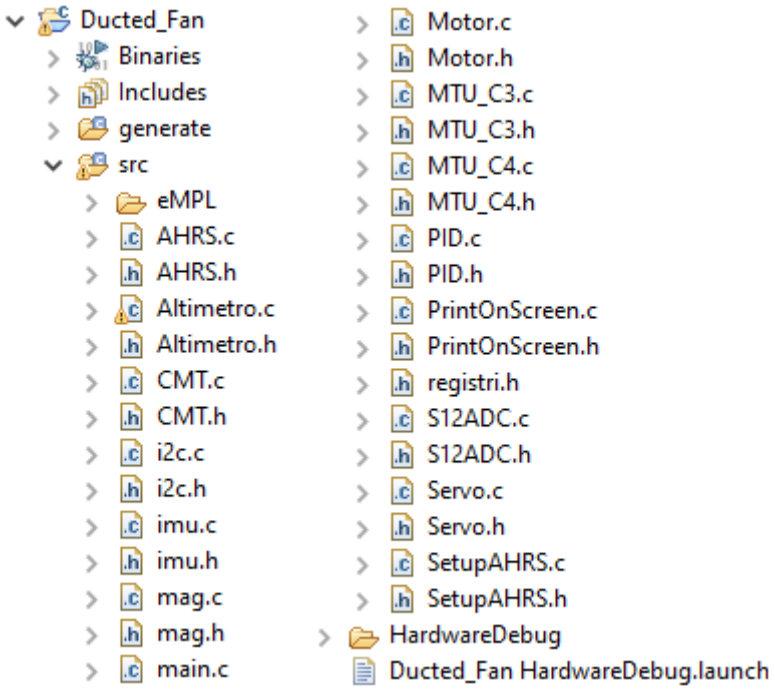


Figura 4.1: Progetto

Come ambiente di sviluppo è stato usato il software e^2 *studio* che viene utilizzato appositamente per sviluppare codice da implementare sui microcontrollori presenti sulle schede RENESAS,

come quella utilizzata per questo progetto. Nella figura 4.1 è mostrata la suddivisione del codice nei rispettivi file, ognuno dei quali si occupa della gestione di un componente specifico del drone. Sono presenti sia file sorgente, .c, che file header, .h, per far si che il codice sia più snello e di facile lettura e comprensione, anche in ottica di sviluppi futuri.
Nel prosieguo di questo capitolo verranno descritte in maniera più dettagliata le parti di codice più importanti per lo sviluppo del progetto

4.1 IDE

Per lo sviluppo completo del software è stato necessario importare i singoli progetti per il controllo dei componenti hardware e fare il *merge* del codice completo. Questo ha portato a errori iniziali in fase di compilazione che verranno descritti brevemente per evitare problemi futuri in caso di sviluppi sul codice.
Essendo errori dovuti a impostazioni su e^2 *studio* è possibile che non possano verificarsi in egual modo per ogni progetto o potrebbero addirittura non presentarsi.

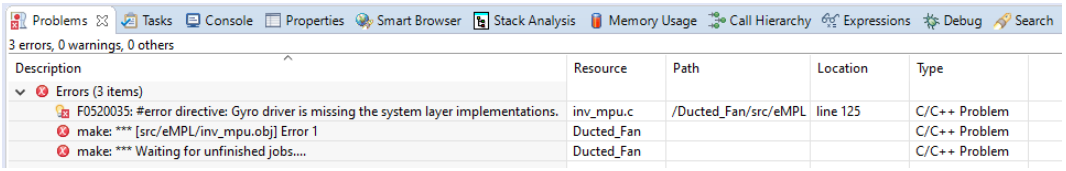


FIGURA 4.2: error directive

L'errore nella figura 4.2 deriva da una mancata definizione, nelle proprietà del progetto, di alcune MACRO necessarie per la corretta compilazione. Per risolvere il problema bisogna seguire i seguenti passaggi:

- Aprire *File*/ *Properties*/ *C/C++ Build*/ *Settings*;
- Nella sezione *Tool Settings* selezionare dall'elenco *Compiler*/ *Source*;
- Nella sezione *Macro definition* inserire le seguenti MACRO
  - EMPL_TARGET_RENESAS;
  - DROTEK_IMU_10DO_V2;
  - MPU6050;

Per applicare le modifiche bisogna, quindi, salvare ed eseguire una nuova compilazione del codice. Le MACRO da aggiungere potrebbero variare a seconda dell'errore specifico da dover risolvere.

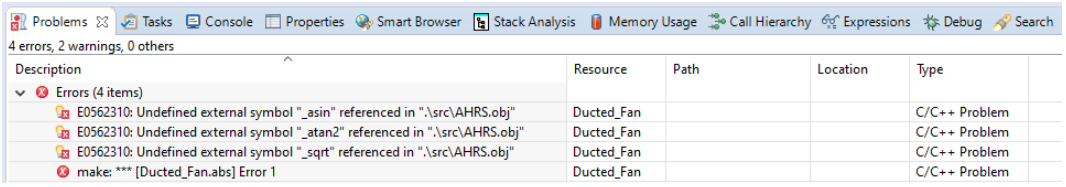


FIGURA 4.3: Undefined external symbol

L'errore nella figura 4.3 è dovuto ad una mancata selezione di librerie matematiche che non vengono quindi inserite in fase di compilazione. Per qualsiasi errore di questo genere, anche con diverse funzioni non rilevate, può essere applicato questo metodo di risoluzione:

- Aprire *File*/ *Properties*/ *C/C++ Build*/ *Settings*;
- Nella sezione *Tool Settings* selezionare dall'elenco *Library Generator*/ *Standard Library*;
- Aggiungere all'elenco di librerie presenti le seguenti:
  - math.h;
  - mathf.h

Così facendo queste due librerie, che si occupano della gestione di funzioni matematiche, verranno aggiunte in fase di compilazione.

4.2 Main

Inizialmente si trovano tutte le inclusioni necessarie per la corretta compilazione del codice, ovvero tutte le librerie utilizzate e gli header file all'interno dei quali sono presenti i prototipi di tutte le funzioni.

```
#include <stdint.h>
#include <stdio.h>
#include <math.h>
#include <mathf.h>
#include <machine.h>
#include "platform.h"
#include "CMT.h"
#include "PID.h"
#include "Altimetro.h"
#include "Motor.h"
#include "SetupAHRS.h"
#include "i2c.h"
#include "PrintOnScreen.h"
#include "Servo.h"
#include "S12ADC.h"
#include "r_switches.h"
```

FIGURA 4.4: Inclusioni

Successivamente vengono dichiarate tutte le variabili globali presenti all'interno del progetto e i prototipi delle funzioni utilizzate all'interno del file che, per motivi di compilazione, non è stato possibile dichiarare all'interno di specifici header file. Una volta che tutto ciò di cui si ha bisogno è stato dichiarato si procede alla scrittura della funzione principale andando ad implementare il codice per il controllo del ducted fan.

Si procede con l'inizializzazione di tutti i dispositivi hardware che verranno poi utilizzati per il controllo del drone. Queste operazioni sono necessarie affinché, prima di ogni utilizzo del drone, tutti i dispositivi tornino al setup iniziale, che permette l'avvio del drone stesso in maniera sicura. Nelle prime fasi un ruolo cruciale è dato dall'utilizzo degli switch presenti sulla scheda RENESAS che verrà descritto in maniera più approfondita nel paragrafo *Switch*.

Il core del main è costituito dal ciclo while che si occupa di gestire tutte le operazioni di lettura dei sensori e di invio dei dati ai motori e ai servomotori.

Essendo la condizione del ciclo while sempre vera è stato necessario inserire al suo interno una funzione per terminare le operazioni. Questo è stato possibile utilizzando un contatore che, una volta raggiunto il valore impostato da codice, spegne in sicurezza i motori terminando il ciclo.

L'operazione che viene svolta implementando l'attuale codice sulla scheda RENESAS è il raggiungimento di una quota di altezza prefissata nel codice che, grazie all'utilizzo del PID per la gestione dei motori, deve essere raggiunta nel minor tempo possibile. Una volta raggiunta la quota desiderata, il drone rimane in volo finché il ciclo while non è terminato mediante l'utilizzo del contatore che viene descritto precedentemente.

Di seguito viene riportato un diagramma che illustra il funzionamento logico del main.

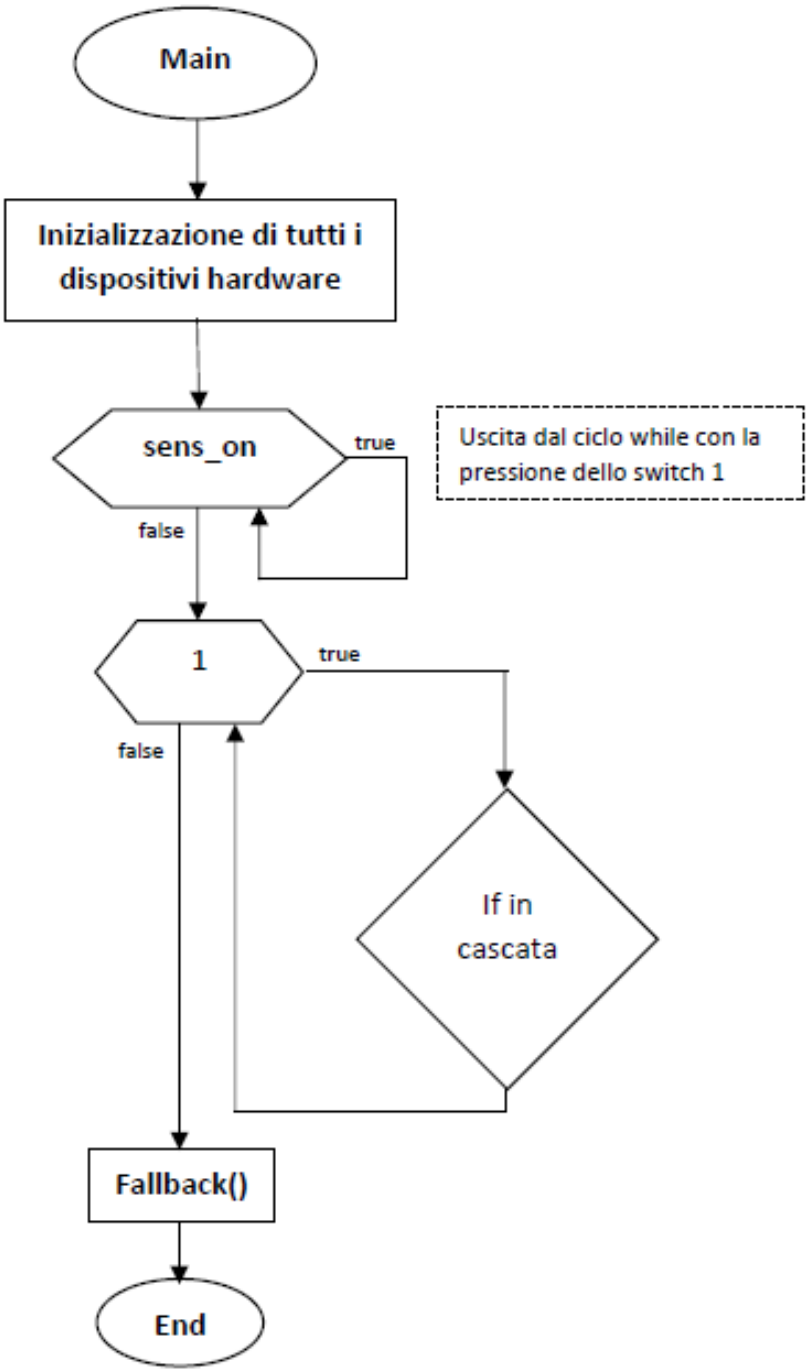


FIGURA 4.5: Diagramma di flusso

L'if in cascata viene riportato nel dettaglio nella pagina successiva.

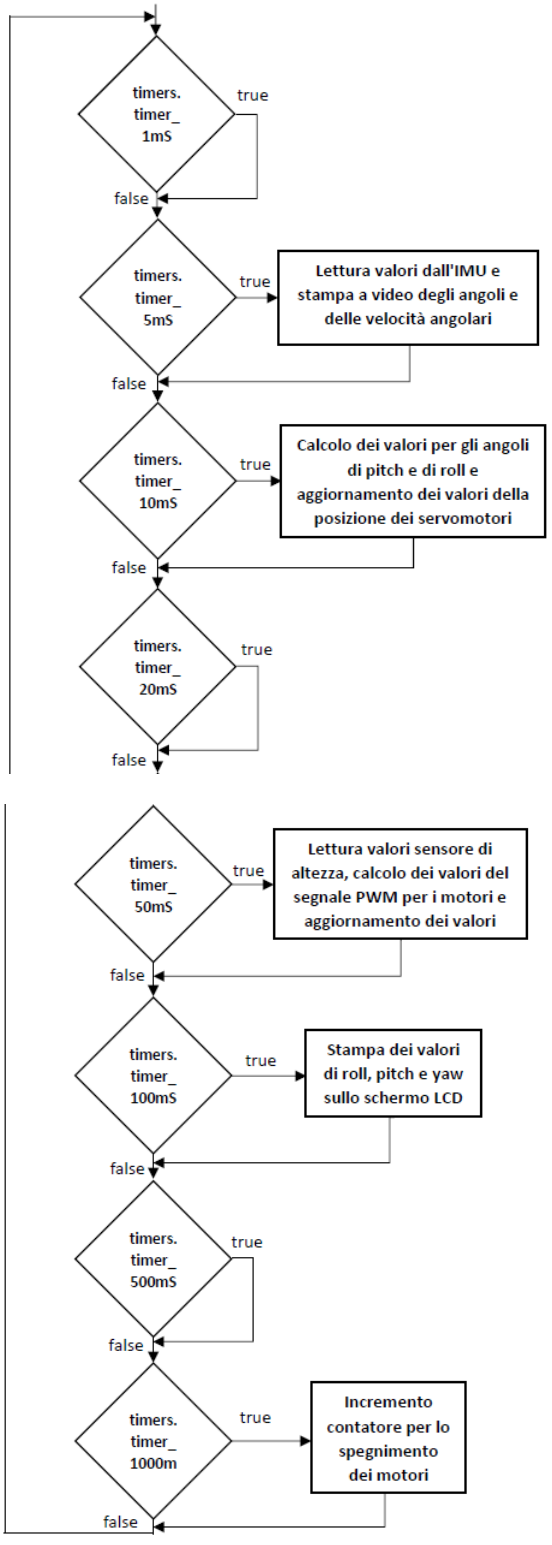


FIGURA 4.6: If in cascata

4.3 Switch

Un ruolo fondamentale per le operazioni di avvio del drone viene svolto dagli switch[1]. Gestiti tramite interrupt inviati alla scheda RENESAS, si occupano di avviare le operazioni principali per il sollevamento in quota del drone.

```
//variabili per regolare accensione motori e sensori
int motore=0;
int dist=1;
int sens_on=1;
```

FIGURA 4.7: Variabili per la gestione degli switch

Nella figura 4.7 viene mostrata la definizione e l'inizializzazione delle variabili utilizzate per il controllo degli switch.

Quando viene alimentata la scheda RENESAS montata a bordo del drone, il software caricato al suo interno viene avviato e le operazioni di inizializzazione descritte nel paragrafo *Main.c* vengono eseguite. Prima di proseguire con l'avvio del ciclo while principale, il software rimane in attesa dell'accensione di tutti i sensori. Questa operazione avviene quando, dopo la pressione dello switch 1, viene generato un segnale di interrupt che verrà gestito dalla scheda RENESAS impostando la variabile $sens\_on = 0$. Questo fa si che l'esecuzione del software possa proseguire entrando nel ciclo while principale.

All'interno del ciclo inizierà l'acquisizione dei dati da parte dei sensori, ma i motori saranno alimentati dal segnale PWM che riesce solamente a tenerli armati, di conseguenza il drone non potrà sollevarsi per raggiungere la posizione desiderata. Anche l'avvio dei motori viene gestito tramite la pressione di uno switch dedicato e si divide in due fasi.

Nella prima, dopo la pressione dello switch 2, viene generato un nuovo segnale di interrupt che verrà, anch'esso, gestito dalla scheda RENESAS impostando la variabile $motore = 1$. Questo fa si che il segnale PWM inviato ai motori sarà uguale al valore necessario a far girare i motori alla velocità minima. Questo è possibile andando ad utilizzare come valore di quota da raggiungere quello letto dal sensore di altezza, in questo modo il PID dei motori rileva come errore, ovvero la differenza tra riferimento e lettura, un valore pari a zero e fornisce in uscita un valore che, dopo opportune conversioni, è dato in ingresso alla funzione *Motor_Write_up*, che verrà descritta più nel dettaglio nel paragrafo *Motori*, che si occupa di aggiornare il segnale di ingresso che gestisce la rotazione dei motori. Una seconda pressione dello switch 2 riporta i motori ad essere

[1] Si trovano alla sinistra dello schermo LCD presente sulla scheda RENESAS

alimentati con un segnale che permette loro di rimanere armati ma che non è sufficiente per farli ruotare alla velocità minima.

Nella seconda fase, che può essere avviata con la pressione dello switch 3, a seguito del segnale di interrupt, la scheda RENESAS imposta la variabile $dist = 0$. Questa operazione avviene dopo che il ciclo while presente nella funzione di gestione dell'interrupt, *sw3_callback()*, arriva al termine, in quanto è stato deciso di fornire all'operatore che effettua la pressione dello switch del tempo per distanziarsi dal drone, evitando problemi di qualsiasi natura, dalla rottura di parti della struttura all'impatto del drone sull'operatore stesso. A seguito di questa operazione il valore di quota effettivo da raggiungere viene dato come riferimento al PID che, dopo aver effettuato i calcoli necessari, fornirà alla funzione *Motor_Write_up* il valore che gestisce la rotazione dei motori.

Quando queste operazioni sono state svolte nella maniera corretta il drone proverà a raggiungere la quota desiderata nel minor tempo possibile e, una volta raggiunta, cercherà di non discostarsi troppo da essa.

```
//pressione switch1, accensione sensori
void sw1_callback()
{
    sens_on=0;
}

//pressione switch2, accensione motori a velocità minima
void sw2_callback()
{
    if(sens_on==0)
    {
        if(motore==0)
            {
                //Avvio motori al minimo
                motore=1;
            }
            else
            {
                motore=0;
            }
    }
}

//pressione switch3, avvio completo dei motori per raggiungimento quota impostata
void sw3_callback()
{
    if(motore==1)
    {
        /* ciclo per ritardare di qualche secondo l'avvio dei motori,
           aggiunto per permettere all'operatore di allontanarsi dal drone
           e mettersi in sicurezza */
        for(int i=0; i<10000000; i++);
        dist=0;
    }
}
```

FIGURA 4.8: Funzioni per la gestione degli interrupt

La pressione degli switch deve avvenire nell'ordine descritto precedentemente ma, qualora questo non dovesse avvenire, il codice è stato implementato in modo tale che non ci siano problemi dovuti ad un ordine errato di pressione.

4.4 Timer

La temporizzazione del ciclo while principale viene gestita dalla scheda RENESAS mediante l'utilizzo del timer presente al suo interno che, una volta inizializzato, è in grado di gestire le operazioni a seconda dell'istante di tempo in cui devono essere eseguite.

```
#pragma interrupt (CMT_isr(vect = VECT(CMT0, CMI0)))
static void CMT_isr(void)
{
    general_timer_mS++;
    timers.timer_1mS = 1;
    if (!(general_timer_mS % 5)) {
        timers.timer_5mS = 1;
        if (!(general_timer_mS % 10)) {
            timers.timer_10mS = 1;
            if (!(general_timer_mS % 20)) {
                timers.timer_20mS = 1;
            }
            if (!(general_timer_mS % 50)) {
                timers.timer_50mS = 1;
                if (!(general_timer_mS % 100)) {
                    timers.timer_100mS = 1;
                    if (!(general_timer_mS % 500)) {
                        timers.timer_500mS = 1;
                        if (!(general_timer_mS % 1000)) {
                            timers.timer_1000mS = 1;
                            if (!(general_timer_mS % 2000)) {
                                timers.timer_2000mS = 1;
                                general_timer_mS = 0;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

FIGURA 4.9: Definizione del timer

Nella figura 4.9 è definita, all'interno del file *CMT.c*, la funzione che si occupa di gestire le funzionalità del timer. Utilizzando la medesima struttura di if in cascata all'interno del main e impostando le condizioni per la verifica del raggiungimento di un certo istante di tempo, è possibile gestire nella maniera corretta tutte le funzioni necessarie per l'acquisizione dei dati da parte dei sensori, indispensabili per la gestione dei motori e dei servomotori.

```
struct timerClocks
{
    uint8_t timer_1mS;
    uint8_t timer_5mS;
    uint8_t timer_10mS;
    uint8_t timer_20mS;
    uint8_t timer_50mS;
    uint8_t timer_100mS;
    uint8_t timer_500mS;
    uint8_t timer_1000mS;
    uint8_t timer_2000mS;
    uint8_t timer_4000mS;
};
```

FIGURA 4.10: struct timerClocks

Per la verifica delle condizioni temporali sopracitate, viene utilizzata la struttura nella figura 4.10, dichiarata nel file *CMT.h*.

```
// Structure containing timer flags
extern struct timerClocks timers;
```

FIGURA 4.11: Definizione struttura

Definendo all'interno del main la struttura *timers* è possibile utilizzare i campi presenti nella figura 4.10 come requisiti per la verifica delle temporizzazioni impostando come condizioni degli if la seguente: $timers.timer\_ * mS$[2].
É fondamentale, quindi, stabilire le priorità di esecuzione delle singole funzioni che sono ripartite nel seguente modo:

- acquisizione dei dati dall'IMU e stampa sullo schermo LCD dei valori ogni 5ms;
- calcolo dei valori da impostare per i flap che gestiscono gli angoli di *pitch* e di *roll* ogni 10ms;
- acquisizione dei valori dal sensore di altezza e calcolo del segnale PWM da impostare sui motori ogni 50ms;
- stampa sullo schermo LCD degli angoli di *roll*, *pitch* e *yaw* ogni 100ms;
- incremento del contatore che si occupa dello spegnimento dei motori dopo il tempo prestabilito ogni 1000ms.

[2]Il valore * va sostituito con il relativo valore di tempo

Tutte le funzionalità sono gestite mediante l'utilizzo di interrupt ogni qualvolta la condizione di temporizzazione risulti verificata, ovvero quando è passato un lasso di tempo necessario a ripetere le operazioni sopracitate. Ogni interrupt viene gestito richiamando la funzione $Callback\_ * ms()$[3] ad eccezione dell'acquisizione dei valori dell'IMU con successiva stampa sullo schermo LCD, che vengono eseguite direttamente all'interno dell'if senza passare per una funzione esterna al main, come avviene in tutti gli altri casi. É stato optato per questa soluzione poiché la stampa a video è onerosa in termini di tempi, di conseguenza, essendo la lettura dell'IMU un passaggio fondamentale per la stabilità in volo del drone, si è preferito evitare ulteriori perdite di tempo che potevano derivare dall'utilizzo di una funzione di Callback.
Per ottimizzare i tempi è possibile, quando si è certi che le misure acquisite sono corrette, eliminare le stampe sullo schermo LCD, rendendo tutto il processo di gestione del drone molto più reattivo nelle operazioni da svolgere.

4.5 PID

Per verificare il corretto raggiungimento di determinati valori, sia per la gestione dei motori che per i servomotori, è stato implementato un PID a livello software. Preso in ingresso un valore di riferimento da inseguire, mediante l'utilizzo di costanti proporzionali, integrative e derivative opportunamente scelte, è possibile gestire l'uscita per raggiungere nel minor tempo possibile il valore desiderato. All'interno del codice è presente sia un PID per la gestione dei motori, sia uno per la gestione dei servomotori.
Il vantaggio che si ha nell'utilizzo del PID per il controllo del sistema, è costituito dall'utilizzo della medesima struttura in entrambe le situazione, variando solamente i dati che la funzione prende in ingresso.

```
typedef struct
{
    float kp, kd, ki;
    float dt;
    float lastError;
    float ITerm;
    float outMax;
    float outMin;
} PID_config;
```

Figura 4.12: Dichiarazione struttura PID

[3] Il valore * va sostituito con il relativo valore di tempo

Nella figura 4.12 è dichiarata, all'interno del file *PID.h* la struttura del PID.

```
/* Create PID structure used for PID properties */
PID_config z_axis_PID;
PID_config Pitch_PID;
PID_config Roll_PID;
```

FIGURA 4.13: Definizione strutture

Per differenziare i PID utilizzati per la gestione degli angoli di *pitch*, di *roll* e per la quota, è necessario definire tre strutture differenti all'interno del main, come riportato nella figura 4.13.

```
*Function name: PID_Compute
float PID_Compute(float input, float setPoint, PID_config* conf);
```

FIGURA 4.14: Dichiarazione $PID\_Compute$

Nella figura 4.14 viene mostrata la dichiarazione, all'interno del file *PID.h*, della funzione utilizzata in tutti e tre i casi sopracitati. Prende in ingresso le variabili *input*, *setPoint* e la struttura *conf* che, a seconda della funzionalità che avranno i PID, assumeranno valori differenti.
Quando è necessario il controllo sull'angolo di *pitch*, la funzione prende come variabile *input* il valore letto dall'IMU dell'angolo di *pitch*; il valore associato alla variabile *setPoint* sarà pari a 0, poiché l'obiettivo è quello di riportare il flap, che gestisce le variazioni di angolo, nella posizione di partenza, ovvero perpendicolare al terreno e con un angolo di 0° rispetto all'asse verticale del drone. La struttura *conf* che viene passata in ingresso sarà *Pitch_PID*, definita nella figura 4.13.
Quando il PID è utilizzato per la gestione dell'angolo di *roll*, alla variabile *input* verrà associato il valore letto dall'IMU dell'angolo di *roll*; la variabile *setPoint* sarà pari a 0 poiché il principio di funzionamento di entrambi i flap è il medesimo, ovvero riportarsi perpendicolari al terreno per fornire stabilità al drone; la struttura *conf* che viene passata alla funzioni è *Roll_PID*, definita nella figura 4.13.
Quando il PID è usato per il controllo di quota, invece, la variabile *input* assume il valore della distanza dal terreno letta dal sensore di altezza, la variabile *setPoint* sarà uguale al valore di quota da raggiungere prestabilito nelle definizioni iniziali del codice. La struttura *conf* avrà in ingresso la struttura z_axis_PID, definita nella figura 4.13.

```
float PID_Compute(float input, float setPoint, PID_config* conf)
{
	/*Compute all the working error variables*/
	float error = setPoint - input;
	conf->ITerm += (error * conf->dt)* conf->ki;
	if((conf->ITerm) > conf->outMax) conf->ITerm = conf->outMax;
	else if((conf->ITerm) < conf->outMin) conf->ITerm = conf->outMin;
	float dInput = (error - conf->lastError) / conf->dt;

	/*Compute PID Output*/
	float output = (conf->kp * error) + (conf->ITerm) + (conf->kd * dInput);

	if(output > conf->outMax) output = conf->outMax;
	else if(output < conf->outMin) output = conf->outMin;

	/*Remember some variables for next time*/
	conf->lastError = error;
	return output;
}
```

FIGURA 4.15: $PID\_Compute$

Nella figura 4.15 viene riportato il codice utilizzato per lo sviluppo software del PID. A seconda dei valori che la funzione riceve in ingresso, descritti precedentemente, genera un'opportuna uscita che permette di raggiungere il valore desiderato.

Quando il PID è utilizzato per il controllo degli angoli di *pitch* e di *roll* l'uscita viene fornita in ingresso alla funzione $Servo\_Write\_deg$, che si occupa di aggiornare i valori della posizione dei servomotori e, di conseguenza, dei flap corrispondenti.

Nel controllo di quota, l'uscita del PID, viene fornita in ingresso alla funzione $Motor\_Write\_up$ che imposta il nuovo valore per la rotazione dei motori che andranno, poi, a generare la spinta sufficiente per far si che il drone raggiunga la quota desiderata.

Per un utilizzo ottimale dei PID è necessario tararli nella maniera ottimale, ovvero in modo tale che i tempi di riposta siano i più brevi possibili per arrivare a regime, facendo si che la risposta sia stabile e non produca troppe oscillazioni attorno al valore desiderato. É necessario, quindi, assegnare i valori alle costanti K_p, K_i e K_d che sono, rispettivamente, il termine proporzionale, integrativo e derivativo, utilizzati per il calcolo dell'uscita del PID. É consigliabile effettuare la taratura dei PID in regime di simulazione del sistema che, però, esula dallo sviluppo di questa tesi, per la quale sono stati utilizzati valori precedentemente stabiliti.

4.6 Motori

Il funzionamento dei motori è gestito dai file *Motor.h* e *Motor.c*. All'interno del primo si trovano le dichiarazioni di tutte le costanti e le funzioni che vengono utilizzate per svolgere tutte le azioni necessarie per far si che il motore produca, insieme alle eliche, la spinta necessaria per far si che il drone raggiunga la quota stabilita. All'interno del secondo si trovano le definizioni delle funzioni con il relativo sviluppo del codice.

```
/* Initialize motors */
Motors_Init();

/* Turn on motors relay */
Motors_On();

/* Send arm signal to motors */
Motor_Arm(MOTOR_UPPER);
Motor_Arm(MOTOR_BOTTOM);
```

FIGURA 4.16: Inizializzazione dei motori

Come tutti i dispositivi hardware utilizzati all'interno del progetto, anche per i motori è necessaria un'inizializzazione che deve essere eseguita prima di ogni utilizzo, nella figura 4.16 vengono riportate le funzioni utilizzate. $Motors\_Init()$ si occupa di inizializzare tutti i registri presenti all'interno della scheda RENESAS che si occupa della gestione dei segnali che vengono inviati in ingresso al motore, $Motors\_On()$ avvia il conteggio del timer che gestisce la temporizzazione di tutte le operazioni effettuate sui motori per poter lavorare correttamente sui registri al suo interno, infine, $Motor\_Arm()$ gestisce l'armamento dei motori, operazione descritta nel capitolo 3, *Architettura Harware*, paragrafo *Motori ed ESC*, insieme a tutte le procedure di inizializzazione. Per qualsiasi operazione effettuata sui motori è necessario distinguere su quale dei due motori si sta lavorando, per questo sono state introdotte due costanti, *MOTOR_UPPER* e *MOTOR_BOTTOM*, che identificano, rispettivamente, il motore superiore e il motore inferiore. Una volta che le inizializzazioni di tutti i dispositivi hardware sono state effettuate ed il codice entra all'interno del ciclo while principale, i motori dovranno variare la loro potenza per raggiungere la quota prestabilita. Queste operazioni vengono svolte dalla funzione *Motor_Write_up()* alla quale vengono passati due valori, il primo per indicare il motore su cui si deve lavorare e il secondo che rappresenta il valore che deve essere impostato nei registri di gestione del motore. Il secondo parametro viene calcolato prendendo come riferimento l'uscita del PID per il controllo di quota e convertendo quel valore, mediante un'opportuna proporzione, in un

corrispondente valore di segnale PWM. Il range di valori ammissibili per la gestione dei motori tramite segnale PWM è impostato da codice ed è rappresentato dalle costanti *MOTOR_MIN_UP* e *MOTOR_MAX_UP* i cui valori sono, rispettivamente, 1200 e 1750. Il valore che viene fornito in ingresso alla funzione *Motor_Write_up()* sarà compreso nel range appena descritto e verrà impostato nei registri della scheda RENESAS che erogherà al motore il segnale PWM corrispondente.

4.7 Servomotori

La gestione software dei servomotori è implementata mediante i file *Servo.h* e *Servo.c*, all'interno dei quali sono dichiarate e definite tutte le funzione necessarie. Il loro utilizzo è strettamente collegato ai flap, in quanto si occupano di gestire la loro posizione durante tutta l'esecuzione del software.

```
void Servos_Init()
{
    /* Set all parameter needed by MTU4 unit and start count */
    Initialize_MTU_C4();
    StartCount_MTU_C4();
    Servo_Write_deg(SERVO_PITCH, START);
    Servo_Write_deg(SERVO_ROLL, START);

}
```

FIGURA 4.17: Inizializzazione dei servomotori

Nella figura 4.17 è riportata la sequenza di inizializzazione dei servomotori, necessaria prima di ogni utilizzo per inizializzare tutti i registri nella maniera corretta e per riportare i flap nella loro posizione iniziale, gestita mediante la costante *START* definita uguale a 0, che fa si che i flap siano perpendicolari al terreno.
Il comportamento dei servomotori e la conseguente posizione dei relativi flap viene gestito mediante la funzione *Servo_Write_deg()* che prende in ingresso due valori, il primo per identificare il servomotore utilizzato e il secondo che rappresenta l'angolo che il flap deve raggiungere. Il primo ingresso può assumere due valori, *SERVO_PITCH* e *SERVO_ROLL* per identificare, rispettivamente, i servomotori che gestiscono la variazione degli angoli di *pitch* e di *roll*. Il secondo ingresso assume il valore di uscita calcolato dal PID di controllo dei servomotori.
Rispetto alla posizione di partenza i flap hanno un range di movimento di $\pm 30°$, che è anche il range di valori che fornisce il PID in uscita. Prima di essere però convertiti in valori che poi

verranno impostati nei registri della scheda RENESAS, è necessario aggiungere un *OFFSET* di 90° che permette di traslare il range di lavoro in quello ammissibile per il servomotore, che va da 0° a 180°. Per questo progetto è però stato limitato tra 60° e 120° per evitare problemi strutturali al drone.

Il segnale PWM utilizzato per la gestione dei servomotori ammette un valore minimo e un valore massimo, rispettivamente pari a 500 e 2500, che indica il valore in micro secondi delle posizioni che il servomotore può assumere. Sono definiti utilizzando le costanti *SERVO_MIN_US* e *SERVO_MAX_US*.

4.8 IMU

Tutte le funzioni di inizializzazione e gestione dell'IMU sono presenti all'interno dei file *IMU.h* e *IMU.c*.

Come tutti i dispositivi hardware è presente un'inizializzazione per favorire il corretto funzionamento del dispositivo prima di ogni nuovo utilizzo. Questa procedura è effettuata dalla funzione *imu_init()*, che abilita il funzionamento del giroscopio e dell'accelerometro presenti all'interno del dispositivo e inizializza tutti i registri dei sensori.

Una volta inizializzato, il dispositivo verrà utilizzato all'interno del ciclo *while* principale come descritto nel paragrafo *Main*.

```
int imu_read(IMU_raw* imu_raw, IMU_sens* imu_sens, IMU_temp* imu_temp){

    short data_acc[3] = {imu_raw->accRoll,imu_raw->accPitch,imu_raw->accYaw};
    mpu_get_accel_reg(data_acc, NULL);
    imu_raw->accRoll = data_acc[0];
    imu_raw->accPitch = data_acc[1];
    imu_raw->accYaw = data_acc[2];

    short data_gyr[3] = {imu_raw->gyrRoll, imu_raw->gyrPitch, imu_raw->gyrYaw};
    mpu_get_gyro_reg(data_gyr, NULL);
    imu_raw->gyrRoll = data_gyr[0];
    imu_raw->gyrPitch = data_gyr[1];
    imu_raw->gyrYaw = data_gyr[2];

    imu_temp->accRoll=imu_raw->accRoll / imu_sens->acc_sens;
    imu_temp->gyrRoll = imu_raw->gyrRoll / imu_sens->gyr_sens * 0.01745329252;
    imu_temp->accPitch = imu_raw->accPitch / imu_sens->acc_sens;
    imu_temp->gyrPitch = imu_raw->gyrPitch / imu_sens->gyr_sens * 0.01745329252;
    imu_temp->accYaw = imu_raw->accYaw / imu_sens->acc_sens;
    imu_temp->gyrYaw = imu_raw->gyrYaw / imu_sens->gyr_sens * 0.01745329252;
    return 0;
}
```

FIGURA 4.18: Acquisizione dati IMU

Nella figura 4.18 vengono riportate le operazioni per l'acquisizione dei dati da parte dell'IMU. All'interno della funzione si trovano tre strutture differenti, *imu_raw* che contiene i dati grezzi letti dall'IMU; *imu_sens* che contiene i dati dei sensori dell'IMU e *imu_temp* che contiene i dati rielaborati delle letture effettuate.
Tutte le operazioni e il funzionamento del dispositivo vengono descritte nel capitolo 3, *Architettura hardware*, paragrafo *IMU*.

Capitolo 5

Prove sperimentali

Per verificare il corretto funionamento del software implementato è necessario effettuare delle prove sia sui singoli dispositivi che sul drone completo.
La prima fase di prove riguarda il funzionamento dei singoli componenti hardware per essere sicuri che, una volta montati sulla struttura, rispondano nella maniera corretta. Per il cablaggio sia in fase iniziale di test che in fase finale, si veda l'appendice B.

5.1 Motori

Per effettuare i test iniziali sui motori, non avendo a disposizione le batterie, è stata scelta come fonte di alimentazione un generatore di tensione impostato per erogare una tensione pari a quella delle batterie, ovvero di $14.7V$. Per simulare la variazione di segnale PWM in ingresso ai motori, che nel funzionamento completo è generato dalla scheda RENESAS, è stato utilizzato un generatore di funzione, per erogare un segnale che rispettasse i limiti descritti nel capitolo 3, *Architettura hardware*, paragrafo *Motori ed ESC*. I test effettuati sui motori sono stati svolti per verificare che, a seguito di variazioni di segnale PWM, la velocità di rotazione dei motori assumesse i valori desiderati.
Inoltre sono state verificate le specifiche di funzionamento dell'ESC, per essere certi che il range di valori del segnale PWM per l'armamento e il funzionamento dei motori fosse corretto. Per la verifica sono stati seguite le specifiche del datasheet dell'ESC utilizzato.

5.2 Servomotori

I test sui servomotori sono stati effettuati allo scopo di controllare che rispondessero nella maniera corretta agli input degli angoli che venivano impostati tramite codice. Alimentati tramite generatore di tensione impostato a $6.5V$, è stato implementato un software specifico per il test. Utilizzando gli switch presenti sulla scheda RENESAS è stato possibile associare alla pressione dello switch 1 un movimento orario da parte del servomotore, con la pressione dello switch 2 un movimento antiorario. Una volta verificato il corretto funzionamento si è proceduto in maniera graduale. Il passo successivo è stato quello di inserire dei limiti di angoli massimi da poter raggiungere per simulare, quella che poi, sarebbe stata la condizione di utilizzo sul drone completo. Sono stati nuovamente utilizzati gli swtich in modo tale che a seguito della pressione dello switch 1 il servomotore compiesse un movimento di $+30°$ rispetto alla posizione di partenza, per poter raggiungere l'angolo massimo positivo impostato. Con la pressione dello switch 2 viene effettuato il movimento contrario, ovvero il ritorno alla posizione iniziale e un movimento di $-30°$ rispetto ad essa, in modo da simulare il raggiungimento dell'angolo massimo negativo impostato.
Questi test sono di estrema importanza in quanto il movimento dei servomotori corrisponde alla posizione dei flap collegati ad essi, necessaria per gestire la stabilità del drone in volo.

5.3 Sensore di altezza

Come descritto nel capitolo 3, *Architettura hardware*, paragrafo *Sensore di altezza*, il sensore è dotato di quattro diverse modalità di lettura che devono essere testate per verificare la correttezza delle misure e poter, poi, scegliere la modalità più adatta per i test finali del drone. Dopo aver alimentato, mediante l'utilizzo di un generatore di tensione impostato a $3.3V$, il sensore ed aver collegato i pin per l'acquisizione dei dati alla scheda RENESAS, è sufficiente variare la distanza del sensore da una superficie fissa, ad esempio un muro, e verificare che la lettura eseguita dal sensore, i cui risultati sono riportati sullo schermo LCD della scheda RENESAS, corrisponda con la distanza effettiva dal muro che può essere misurata da un operatore mediante l'utilizzo di un metro.
É necessario testare tutte le modalità di acquisizione dati e verificare che i valori presenti nella tabella 3.1 siano corretti, per poter scegliere in maniera accurata le modalità di lettura in fase finale di test.

5.4 IMU

Per garantire la stabilità in quota del drone è necessario conoscere gli angoli di *pitch* e di *roll*, è perciò fondamentale garantire un corretto funzionamento della lettura dei dati dall'IMU. Per effettuare i test di prova è stato utilizzato un generatore di tensione per l'alimentazione, impostato su $3.3V$, e il collegamento con la scheda RENESAS per i pin *SDA* e *SCL*, come riportato nell'appendice B. La prima prova effettuata è stata una simulazione manuale dei movimenti che avrebbe poi compiuto l'IMU una volta montata a bordo del drone ed è stato verificato che le letture effettuate, i cui valori sono stampati sullo schermo LCD della scheda RENESAS, corrispondessero con le reali inclinazioni dell'IMU. In fase di test si è notato che ad un valore di $\pm 90°$ dell'angolo di *roll* corrispondeva una singolarità del sistema che forniva valori errati di letture. Per questo motivo si è optato, in fase di assemblamento finale, di posizionare l'IMU parallelamente al terreno, così da evitare valori dell'angolo di *roll* troppo elevati che potessero creare problemi nella gestione del drone in quota.

5.5 Avanzamento dei test

Una volta che tutti i dispositivi hardware sono stati testati singolarmente si può procedere con delle prove più complesse. In particolare si è deciso, prima di effettuare i test finali, di svolgere delle prove intermedie per avanzare gradualmente ed evitare errori.
É stata introdotta l'implementazione software del PID, come descritto nel capitolo 4, *Architettura software*, paragrafo *PID*, per verificare il corretto comportamento da parte dei motori e dei servomotori. In particolare è stato verificato che i motori rispondessero correttamente incrementando la propria rotazione nel caso in cui il valore acquisito dal sensore di distanza fosse minore del valore di riferimento da raggiungere, viceversa, diminuendo la rotazione nel caso in cui la distanza di riferimento fosse superata. I primi test sono stati svolti senza eliche per evitare problemi dovuti a comportamenti non consoni da parte del drone.
Per il test sui servomotori è stato verificato che, una volta posizionati i flap nella posizione iniziale, ovvero perpendicolari al terreno, a seguito di letture di angoli diversi da 0 da parte dell'IMU, i servomotori si muovano nel verso opposto a quello di sbilanciamento del drone, per far si che quest'ultimo possa tornare nella posizione iniziale sfruttando il flusso d'aria deviato dai flap. Quando tutte le operazioni vengono svolte correttamente è possibile effettuare dei test con il drone completamente assemblato.

É stata costruita una struttura tubolare in grado di contenere il drone, fornendo stabilità completa sul piano orizzontale, favorendo però libertà di movimento sull'asse verticale, in modo tale che il sistema di raggiungimento di quota potesse essere testato senza doversi preoccupare di stabilizzare il drone in volo. I test sono stati svolti con una quota da raggiungere di circa $20cm$ dal suolo e sono stati eseguiti con successo, provando il corretto sviluppo del software per il controllo di quota.
Il test successivo è stato effettuato per il controllo di assetto in quota del drone. Per effettuare le prove in sicurezza è stato posizionato il drone all'interno di una gabbia di contenimento per evitare risposte errate da parte dei motori che potrebbero portare il drone a movimenti poco consoni. Inoltre è stato vincolato il drone alla gabbia mediante l'utilizzo di cavi di sicurezza che, in caso di pericolo, possono essere tesi per alzare in drone e immobilizzarlo in sicurezza. Non si hanno risposte certe per quanto riguarda il corretto funzionamento del controllo di assetto in quota poiché il numero di test effettuati non è stato sufficiente. Dalle prove effettuate si può, però, auspicare un corretto funzionamento che può essere perfezionato aumentando il numero di test svolti.

Capitolo 6

Conclusioni

Si conclude questo elaborato in cui è stato affrontato il problema dello sviluppo del software per il controllo del volo di un drone a flusso convogliato.
Sono state analizzate tutte le componenti hardware utilizzate e, successivamente, è stato descritto lo sviluppo del software implementato sulla scheda RENESAS per la gestione del drone. In particolare, sono state analizzate nel dettaglio tutte le sezioni di codice importanti per il corretto funzionamento del drone e sono state riportate le linee di codice ritenute cruciali nello sviluppo di parti essenziali del software.
Infine sono state descritte le modalità dei test svolti per verificare il corretto sviluppo del software, descrivendo prima le prove effettuate sui singoli componenti e successivamente i test con il drone completo.
Per sviluppi futuri del progetto si consiglia l'implementazione del calcolo della distanza del drone dal suolo quando, quest'ultimo, subisce variazioni sugli angoli di *pitch* e di *roll*, che in questo momento non vengono considerate, portando così i valori misurati a subire errori dovuti al mancato posizionamento perpendicolare del sensore rispetto al suolo. Questa funzionalità può essere implementata aggiungendo il calcolo della distanza tramite l'utilizzo di funzioni trigonometriche che prendono come argomenti i valori degli angoli calcolati dall'IMU. Inoltre può essere aggiunto lo spegnimento controllato dei motori del drone che permettono, così, il ritorno controllato al suolo, dopo aver raggiunto la quota desiderata. Per ottenere prestazioni più performanti nei tempi di risposta dei motori e dei servomotori, può essere effettuata una nuova taratura dei PID.

Appendice A

Disposizione hardware

Per la corretta distribuzione dei pesi e per far si che il centro di massa sia posizionato sull'asse di rotazione verticale del drone, che coincide con l'asse di rotazione dei motori, è stato necessario posizionare i dispositivi hardware come mostrato in figura:

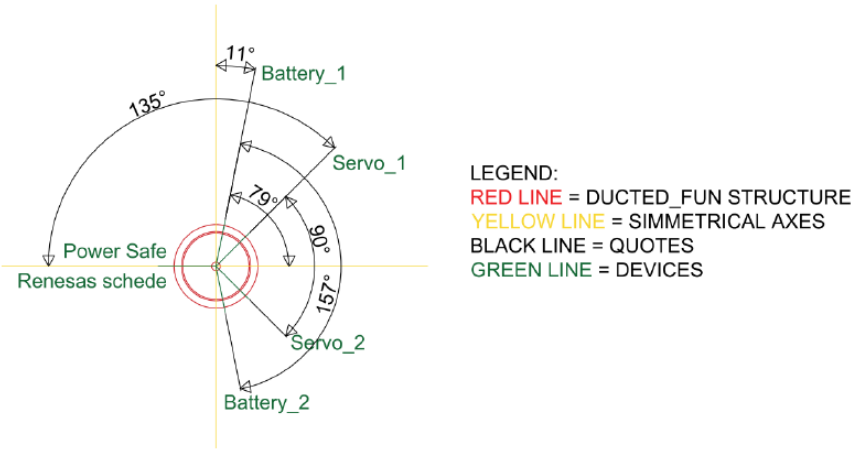


Figura A.1: Posizionamento dispositivi hardware

Nel caso in cui si decidesse di cambiare qualche dispositivo hardware in sviluppi futuri questa disposizione potrebbe subire delle modifiche in quanto è stata calcolata tenendo conto dei pesi dei singoli componenti hardware. Cambiando i componenti potrebbe quindi variare la distribuzione dei pesi e di conseguenza dovranno essere calcolati nuovamente gli angoli corretti. Per farlo si devono uguagliare i pesi della parte sinistra con quelli della parte destre e moltiplicarli per le loro distanze angolari, ovvero le distanze rispetto all'asse verticale.

Appendice B

Cablaggio

Di seguito sono riportati i collegamenti effettuati in fase di cablaggio per tutti i dispositivi hardware.

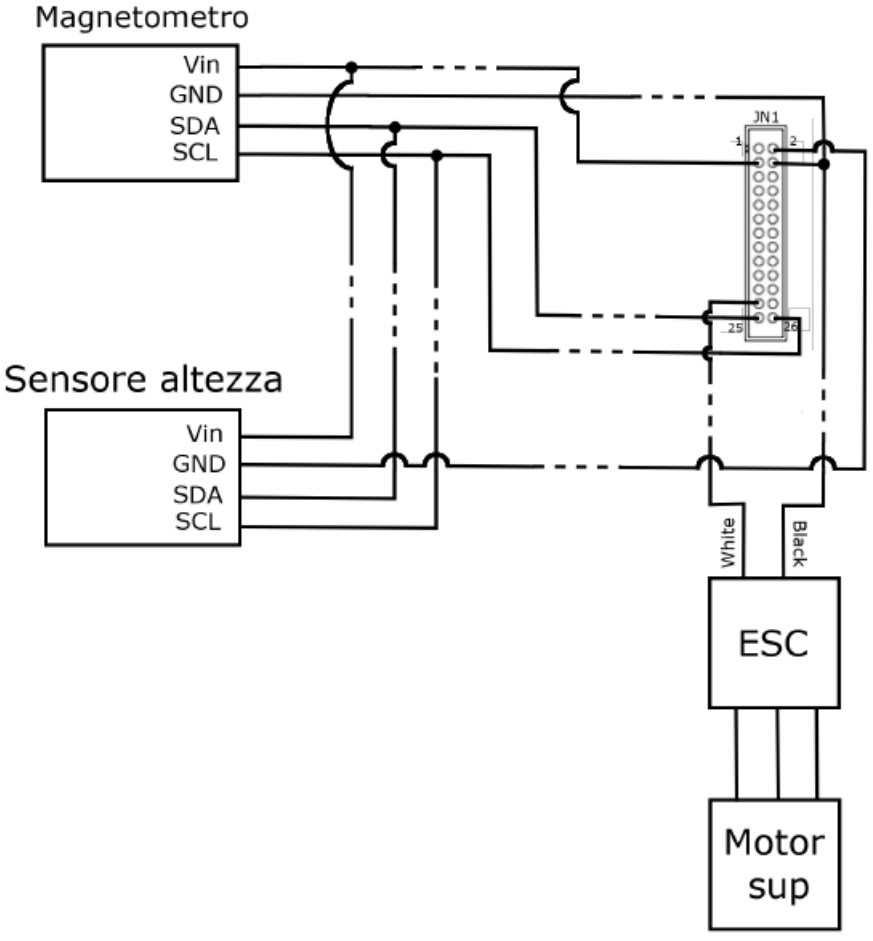


Figura B.1: Cablaggio magnetometro, sensore di altezza e motore superiore

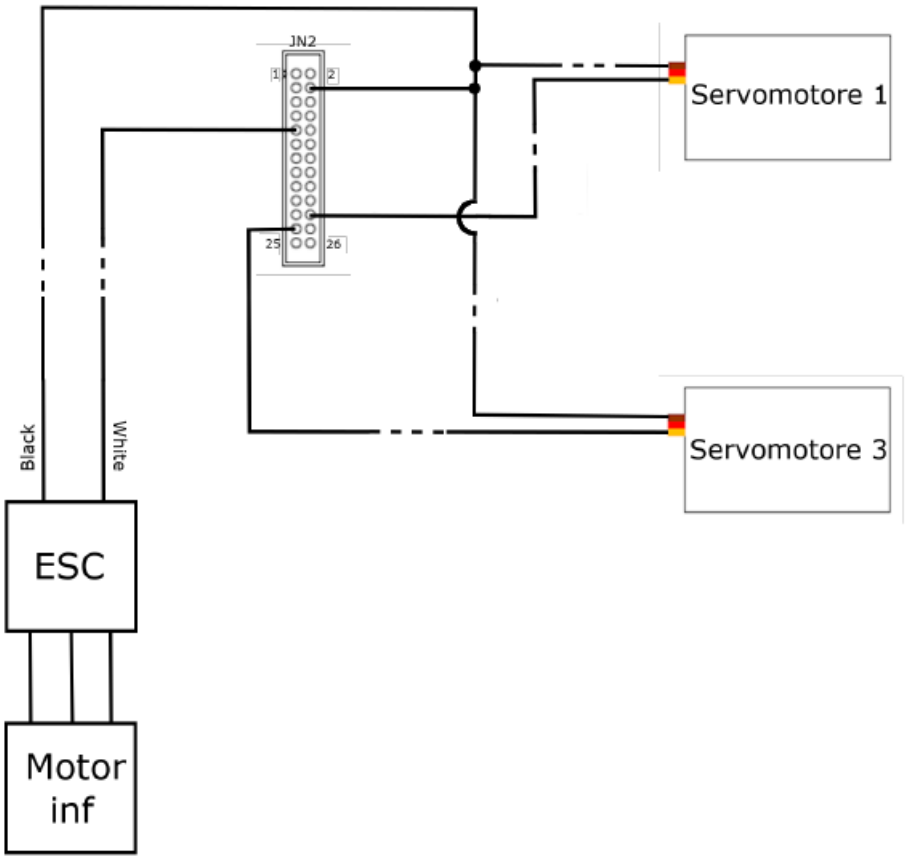


FIGURA B.2: Cablaggio servomotori e motore inferiore

Nella tabella B.1 vengono riportate tutte le connessioni tra i dispositivi hardware e la scheda RENESAS. Si noti che alcuni dispositivi hanno dei pin in comune, quindi per il corretto collegamento sono stati saldati tra di loro i connettori. Le masse di tutti i dispositivi (GND) sono collegate alla massa comune della batteria

Dispositivo	Porta	Colore	JN*/pin
Sensore di altezza	Vin	-	PWR(3.3V)
	GND	-	GND
	SDA	-	JN1/25
	SCL	-	JN1/26
Magnetometro	Vin/3.3V	-	PWR(3.3V)
	GND	-	GND
	SDA	-	JN1/25
	SCL	-	JN1/26
Servmotore1 (pitch)	Vin	Rosso	PWR(6.5V)
	GND	Marrone	GND
	PWM	Arancione	JN2/22
Servmotore3 (roll)	Vin	Rosso	PWR(6.5V)
	GND	Marrone	GND
	PWM	Arancione	JN2/23
Motore sup/ESC1	PWM	Bianco	JN1/23
	GND	Nero	GND
Motore inf/ESC2	PWM	Bianco	JN2/9
	GND	Nero	GND

Tabella B.1: Collegamenti

Bibliografia

[1] Matteo Biagiola. *Modellazione e Simulazione di un Ducted Fan.* versione italiana edition, 2014.

Ringraziamenti

Innanzitutto vorrei ringraziare il Prof. Andrea Bonci per avermi dato la possibilità di svolgere questo tirocinio e per la disponibilità dimostrata nei miei confronti. Ringrazio anche l'Ing. Giuseppe Antonio Scala e l'Ing. Giacomo Nabissi per essersi messi a disposizione ed avermi aiutato nello sviluppo del progetto.

Non smetterò mai di ringraziare la mia famiglia per avermi supportato in ogni momento di questo percorso universitario e non solo.
In particolare ringrazio mio padre, Flavio, un punto di riferimento nella mia vita, che ci ha sempre messo al primo posto in ogni occasione ed è sempre pronto a fare di tutto per la nostra famiglia. Spero di poter diventare deciso e determinato almeno un centesimo di quanto lo è lui. Anche se non te lo dico quasi mai, grazie di tutto.
Ringrazio mia madre, Luana, per aver sempre creduto in me nonostante tutto. Questo traguardo è anche merito tuo, per la dedizione con cui mi sei sempre stata vicino nello studio, e non solo, fin dalle elementari, dove avevo bisogno di controllo costante per la mia bravura innata nel distrarmi in un secondo, forse adesso sono migliorato un po' sotto quell'aspetto. Per avermi ripetuto giorno dopo giorno quanto credi in me, nonostante qualche periodo negativo dal punto di vista universitario in questi anni.
Spero, con questo primo traguardo, di poter ripagare in minima parte tutti i sacrifici che avete sempre fatto per me. Grazie.
Ringrazio mio fratello, Luca, per essere così determinato in qualsiasi cosa tu faccia, dalla più semplice alla più complessa. Grazie a te so che niente è impossibile e che con la giusta grinta tutto si può fare. Grazie per tutte le litigate passate, che si sono sempre trasformate in enormi sorrisi e risate. Grazie per aver sempre trovato la maniera per farmi ridere anche dopo un esame andato male. Sappi che per qualsiasi cosa io per te ci sarò, sempre. Ti voglio bene.
Ringrazio Giacomo, per essere stato, nei miei primi anni universitari, sempre pronto ad aiutarmi in qualsiasi situazione. Per avermi insegnato che dopo qualsiasi bocciatura o difficoltà, l'importante è rialzarsi sempre a testa alta e continuare, proprio come hai fatto tu. Finalmente questo traguardo è stato raggiunto.
Ringrazio Fabrizio, per aver preso ben due lauree prima di me, spronandomi così a cercare di fare il massimo per raggiungerlo. Dopo questa siamo più vicini. Grazie per tutte le risate fatte insieme e per tutti i tagli di capelli gratis. Adesso direi che me ne merito un po'.

Ringrazio Klizia e Julia, per tutti i momenti passati insieme, per le chiamate fatte e per essere così come siete. Stare con voi è sinonimo di felicità, rimanete per sempre così pazze.

Ringrazio tutti i miei amici che in questi anni mi sono stati vicini e che, in un modo o nell'altro, hanno reso tutti i momenti passati con loro dei momenti di gioia. In particolare ringrazio Luca, Loris, Matteo, Schiazza, Valerio, Gianmarco, Irene, Lisa, Marianna, Noemi e Samuele.
Un ringraziamento speciale va ad Andrea, per aver sempre creduto in me, forse in alcuni momenti anche più di quanto non facessi io, questo traguardo è anche merito dei nostri fantastici pomeriggi di "studio" delle superiori. Grazie di tutto.

Ringrazio tutte le persone conosciute in questi anni universitari.
In particolare ringrazio Agostino, prima amico, poi coinquilino, per avermi supportato ma soprattutto sopportato in questi anni, per tutti i momenti fantastici passati insieme e per avermi sempre spronato a fare meglio. Grazie anche per lo splendido disegno per l'abstract della tesi, forse è anche meglio del drone originale.
Ringrazio Alessandro, che con il suo modo di fare diretto, in poco tempo ha saputo prendere un ruolo importante nella mia vita. Se hai bisogno di qualche ripetizione in cucina non esitare a chiedere che sono contento di aiutarti.
Ringrazio Leonardo, prima coinquilino, poi amico, per aver condiviso con me gli ultimi cinque anni della mia vita. Per aver sempre trovato il modo di farmi sorridere anche nei momenti più duri e per aver sopportato in miei silenzi infiniti durante i pasti. Essendo una delle persone con cui ho passato più tempo in questi anni, sono davvero contento di aver condiviso con te questa esperienza di vita.
Ultimo, ma non per importanza, ringrazio Marco, da quella famosa corsa mai fatta insieme sei entrato di prepotenza nella mia vita e non ne sei più uscito. Grazie per tutte le ore di studio e di laboratorio passate insieme, per tutti i pranzi, le cene e le nottate passate da noi. Per tutti i momenti di telepatia condivisi e per capirmi ogni volta senza neanche dover parlare.