

# Progettazione di un drone autonomo e sviluppo dei relativi algoritmi di controllo

Riccardo Pogliacomi

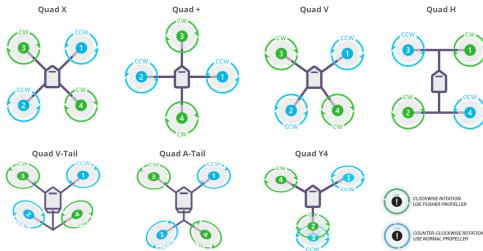
Maggio 2024

## 1 Introduzione

Il progetto descritto in questo documento è nato a seguito della mia passione nell'ambito della teoria del controllo. Infatti, questo drone viene utilizzato come piattaforma di sviluppo per testare algoritmi di controllo.

## 2 Cos'è e come funziona un drone

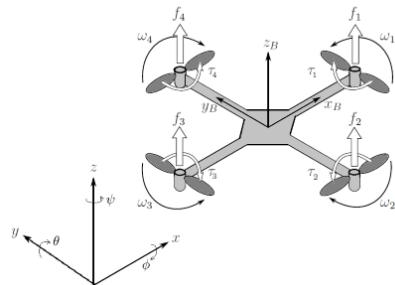
Drone, da definizione non è altro che un veivolo privo di pilota che può essere o non essere comandato a distanza. Esistono vari tipi di drone, nel nostro caso parliamo di quadricottero, che come possiamo capire dal nome si tratta di un veivolo con quattro eliche indipendenti. Tali eliche possono essere disposte in varie configurazioni, la scelta di quest'ultima comporterà anche una diversa strategia di controllo.



**Figura 1:** Possibili configurazioni che il drone può assumere

La disposizione scelta per questo progetto è quella a X che fornisce i giusti compromessi per

agilità e stabilità. Le eliche sono diagonalmente controrotanti in modo tale che si annullino le coppie di forze generate dalla rotazione delle eliche, infatti, se ruotassero tutte nella stessa direzione il drone tenderebbe a rotare attorno al suo asse verticale.



**Figura 2:** Rappresentazione schematica del drone e della relativa terna di riferimento

Per stabilizzare e controllare il drone sfruttiamo i motori indipendenti, infatti per i due assi orizzontali (Roll e Pitch) basterà incrementare i due motori da un lato e decrementare quelli opposti, così da generare un momento angolare sul corpo del drone. Più complesso però è controllare la rotazione sull'asse verticale, infatti dobbiamo fare in modo che le coppie generate dai due motori che ruotano in direzione concorde a quella desiderata, prevalga, per fare ciò, dobbiamo incrementare diagonalmente i motori. Infine la navigazione sarà totalmente autonoma, infatti il drone seguirà un piano di volo, inserito in un software di gestione apposito, composto da punti GPS, tempi di percorrenza ed eventuali soste.

## 3 Design

In questa sezione saranno affrontati gli aspetti relativi alla progettazione sia hardware che software, includendo la scelta dei componenti e quindi il relativo assemblaggio e nella sezione successiva sarà affrontata la parte di sviluppo degli algoritmi.

### 3.1 Scelta dei componenti

La prima cosa da fare quando si decide di sviluppare un drone, ma in generale un qualsiasi progetto, è quella di decidere un applicazione per tale progetto (quale sarà l'impiego definitivo), infatti, partendo da ciò possiamo già capire di che tipo di componenti avremo bisogno.

#### 3.1.1 Frame

La prima cosa da scegliere è il telaio, ovvero dove andremo a montare tutti i nostri componenti. Ne esistono di vari tipi che differiscono per dimensione, peso, materiale costruttivo e quindi le relative caratteristiche meccaniche. Inoltre, la scelta del nostro telaio porta di conseguenza una certa dinamica al drone, un telaio più grande comporterà anche un momento di inerzia maggiore, quindi, se si vuole un drone agile e veloce non si sceglierà un telaio di grosse dimensioni. Nel caso preso in causa in questo documento è stato scelto un telaio abbastanza grande (distanza diagonale da motore a motore di 45 cm), tale scelta è stata fatta per andare incontro alle nostre esigenze, infatti volevamo una piattaforma che permettesse di portare un payload abbastanza elevato visto che oltre ai computer di bordo e relativi sensori, in futuro aggiungeremo un raspberry che in accoppiata con una videocamera si occuperà degli algoritmi computer vision per la detezione di ostacoli.



Figura 3: F450 frame

#### 3.1.2 Motori

Una volta scelto il frame, è il momento di scegliere i motori. Esistono vari tipi di motori, quelli utilizzati maggiormente in questo ambito sono i BLDC, ovvero motori brushless a corrente continua, tali motori vengono classificati in base ai KV, il quale è un parametro che indica l'aumento teorico degli RPM(Revolute per minute) all'aumentare della tensione. Ad un motore con KV alto andrà abbinata un'elica di piccole dimensioni, in quanto se si accoppiasse un'elica di grandi dimensioni, il motore dovrà fornire una coppia molto maggiore e per fare ciò richiederà una maggiore corrente. Questo sovraccarico porterà inevitabilmente il motore a surriscaldarsi e a bruciarsi, in quanto il rivestimento sulla bobina inizierà a fondere e causerà cortocircuiti elettrici nel motore. Quindi un drone di grandi dimensioni, come quello preso in considerazione in questo documento, necessiterà di eliche anch'esse di grandi dimensioni e di conseguenza, di motori con KV bassi(nel nostro caso 1000KV).



Figura 4: BLDC Motor 1000kv (a2212/13t)

### 3.1.3 ESC

Per controllare i motori sono necessari degli specifici driver, ovvero gli ESC (Electronic Speed Controller), questo componente è responsabile della regolazione della velocità del motore attraverso la gestione delle tre fasi del motore. Il controllo di questo componente è gestito attraverso un segnale PWM (Pulse Width Modulation), che gli viene direttamente fornito dal microcontrollore. In questo documento non discuteremo direttamente del funzionamento della PWM, ma per il controllo dei motori basta sapere che ad un segnale PWM di 1000 microsecondi equivale il motore fermo, mentre un segnale a 2000 microsecondi porta il motore al massimo degli RPM possibili. Ovviamente, mantenere il motore al massimo per un lungo periodo di tempo, può portare a danneggiamenti al motore, infatti per sicurezza noi manterremo un margine di 150 microsecondi (max = 1850 microsecondi).

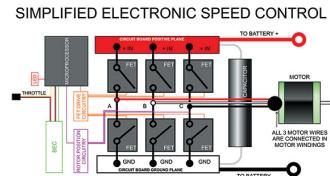
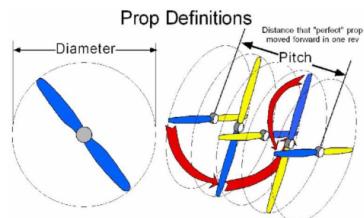


Figura 5: Schema interno di un ESC

### 3.1.4 Eliche

L'elica è il componente che permette di trasformare la velocità di rotazione del motore in spinta, i parametri più importanti che descrivono le eliche sono due, diametro e passo: Il diametro, come si può intuire, è il diametro dell'elica vista frontalmente, mentre il passo è la grandezza che esprime la distanza percorsa da un'elica in un giro completo.



### 3.1.5 Microcontrollore

Per eseguire gli algoritmi di controllo che permettono la navigazione autonoma, c'è bisogno di un microcontrollore (se non un vero e proprio computer). Per questo progetto è stata scelta una Teensy 4.1, equipaggiata con una ARM Cortex-M7, e programmato con il framework di Arduino.

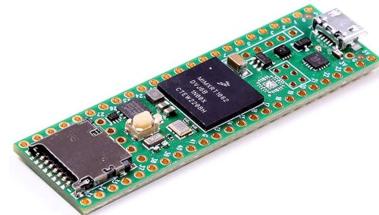


Figura 6: Teensy 4.1

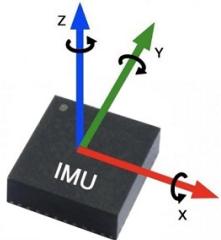
-Caratteristiche generali:

- ARM Cortex-M7 at 600MHz
- 1024K RAM
- 128Mb Flash
- 3 CAN Bus
- 3 SPI
- 3 I2C
- 7 Serial
- 31 PWM pins

### 3.1.6 Sensori

Per riuscire a stimare in modo accurato lo stato del drone, c'è bisogno di svariati sensori. Come primo sistema di localizzazione (visto che vogliamo che la navigazione si basi su punti GPS), ci servirà un modulo GPS. Di tale modulo ne esistono di svariati tipi, molto importante da tenere in controllo è la sensibilità, infatti, ogni modulo ha il proprio errore che varia dagli 8 metri fino a 2 centimetri (nei casi dei sistemi RTK). Ogni modulo inoltre si interfaccia con il microcontrollore attraverso protocolli diversi, quindi in fase di scelta bisogna controllare che tale protocollo sia supportato dall'MCU (MicroController Unit). Il sensore più importante presente sul quadricottero è sicuramente l'IMU (Inertial Measurement Unit), la quale ci permette di misurare l'attitude del drone attraverso la misurazione della velocità con cui cambia l'angolo sui tre assi e l'accelerazione

lineare anch'essa sui tre assi. Questa categoria di sensori, misurando i tre assi sia nell'ambito rotazionale che in quello lineare, viene chiamata IMU 6DOF (6 Degrees Of Freedom), ovvero IMU a sei gradi di libertà. Avendo a bordo un accelerometro, l'IMU deve essere montata in modo tale che non venga influenzata dalle vibrazioni generate dai motori, per fare ciò, il sensore non è stato direttamente fissato al modulo dell'avionica ma sono stati posizionati dei pad in silicone che si occupano di assorbire le vibrazioni. Inoltre, vista l'importanza di questo sensore (l'algoritmo di stabilizzazione si basa interamente su questo sensore), sarebbe bene predisporre della ridondanza, attraverso l'aggiunta di più copie del sensore, così che il computer di volo possa passare da un sensore all'altro in caso di necessità, senza mai compromettere la sicurezza del volo.



**Figura 7:** Funzionamento Inertial Measurement Unit

Per misurare l'altezza dal suolo abbiamo due opzioni a bordo, la prima è quella di usare direttamente il GPS, che oltre alla latitudine e alla longitudine ci permette di misurare anche l'altitudine, questo sistema però ha due problematiche principali, ovvero: innaffidabilità del valore misurato (in quanto il gps ha un alto errore) e la lentezza nel ricevere un nuovo valore. Proprio per questi due motivi abbiamo deciso di montare un nuovo sensore, ovvero un LIDAR, il quale misura l'altezza dal suolo attraverso un impulso laser, questo sistema ci permette di avere un accuratezza nell'ordine del centimetro che ci permette di effettuare atterraggi precisi e autonomi.



**Figura 8:** Lidar TF-mini luna

-Sensori scelti per questo drone:

- GPS: Beitian BN-280
- IMU: MPU6050
- LIDAR: TF-mini luna
- Barometer: BMP180
- Magnetometer: GY-271

### 3.1.7 Comunicazione a terra

Per finire ci serve un modo per comunicare a terra, infatti il drone per l'intera durata del volo comunica con la ground station per mandare la telemetria e per ricevere i comandi, come ad esempio quello che fa partire il volo o quello che fa abortire il volo. Per fare ciò, esistono svariati metodi come ad esempio l'utilizzo delle onde radio oppure quello del wi-fi. Nel nostro caso abbiamo scelto dei moduli a 2.4 Ghz, uno montato sul drone e uno invece collegato a terra ad un arduino UNO, il quale a sua volta è connesso tramite seriale ad un computer dove c'è in esecuzione un software sviluppato da noi che ci permette di visualizzare la telemetria e di mandare i comandi.



**Figura 9:** Nrf24l01 + PA + LNA

## 4 Stima e controllo

In questa sezione affronteremo i problemi della stima dell'attitudine (specialmente ci occuperemo del filtraggio dell'IMU) e controllo del sistema.

### 4.1 Leggere e filtrare i dati dei sensori

L'IMU ci mette a disposizione due dispositivi: un giroscopio e un accelerometro. Se noi provassimo a derivare attitudine del sistema solamente integrando i valori letti dal giroscopio (vanno integrati in quanto il giroscopio legge le velocità rotazionali attorno agli assi), incapceremmo nel problema del gyro bias (problema presente in maniera più o meno visibile in tutti i giroscopi). Il gyro bias è l'output del giroscopio quando stazionario, infatti il sensore ha un offset rispetto al valore reale che dovrebbe rilevare. Se invece provassimo a misurare l'angolo attraverso l'accelerometro riscontreremmo una misurazione molto precisa ma inutilizzabile in quanto molto influenzabile dalle vibrazioni. Le formule (1) e (2) servono per calcolare, attraverso l'accelerometro, rispettivamente l'angolo relativo all'asse X (roll) e Y (pitch).

$$\psi_a = \arctan \left( \frac{a_y}{\sqrt{a_x^2 + a_z^2}} \right) \quad (1)$$

$$\theta_a = \arctan \left( \frac{a_x}{\sqrt{a_y^2 + a_z^2}} \right) \quad (2)$$

Quindi sappiamo che il giroscopio ha dei vantaggi nel breve periodo, ma presenta comunque un influente gyro bias, mentre l'accelerometro ha di positivo il fatto che non soffre di nessun tipo di bias ma ha lo svantaggio della forte rumorosità del segnale. Per stimare lo stato rotazionale del drone, ci serve un modo per unire i vantaggi dei vari sensori, per fare ciò, possiamo usare le tecniche della Sensor Fusion, in particolare per questa applicazione le scelte migliori sono due, o il famosissimo filtro di Kalman o il complementary filter. Del

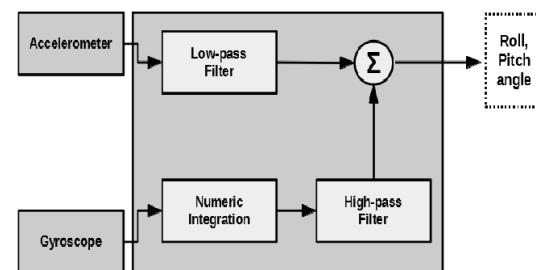
filtro di Kalman non discuteremo in questo documento, visto che è molto complesso e per la totale comprensione richiede elementi di matematica universitaria, inoltre un filtro di kalman richiede anche una discreta potenza computazionale. Mentre il complementary filter, porta dei buoni risultati e per un microcontrollore è molto più facile da eseguire

#### 4.1.1 Complementary filter

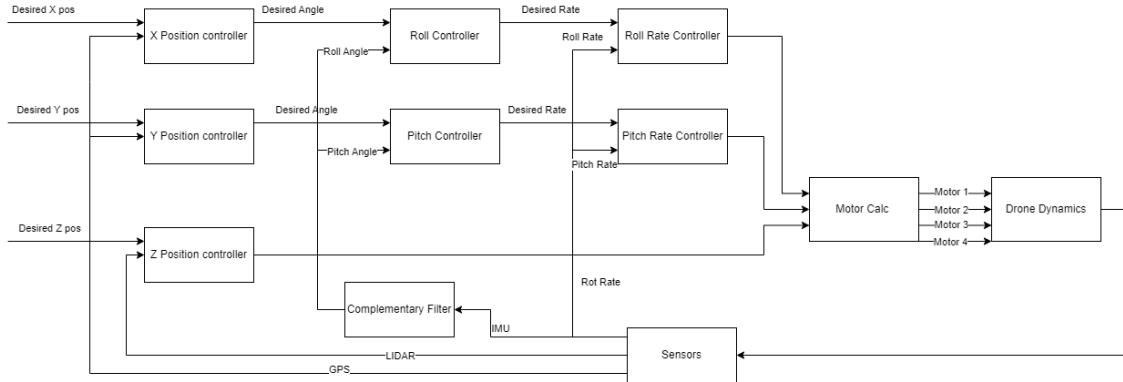
Per stimare lo stato del drone abbiamo bisogno di sfruttare l'accuratezza nel breve periodo fornita dal giroscopio e la stabilità nel lungo periodo dell'accelerometro. Il complementary filter ci permette di fare ciò, poiché applica un low-pass filter sull'accelerometro e un high-pass filter sul giroscopio.

$$\beta = (1 - \alpha) * (\beta_{prev} + d\beta * dt) + \alpha * \beta_{acc} \quad (3)$$

L'equazione (3) è l'implementazione del complementary filter, dove  $\beta$  è l'angolo finale filtrato,  $\beta_{prev}$  è l'angolo filtrato precedentemente, dt è il tempo passato dall'esecuzione del filtro precedente,  $d\beta$  è la velocità angolare e infine  $\alpha$  è un coefficiente che decide quanto affidarsi all'accelerometro. Ovviamente questa equazione va risolta per ogni angolo orizzontale, quindi roll ( $\psi$ ) e pitch ( $\theta$ ).



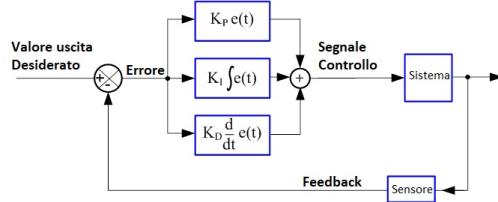
**Figura 10:** Schema complementary filter



**Figura 11:** Schema di controllo del drone

## 4.2 Algoritmi di controllo

In questa sezione discuteremo delle scelte fatte nell'ambito del controllo, tutto lo schema di controllo relativo al drone è rappresentato nella figura 11. Per questa prima versione tutti i controller internamente sono costituiti da controller PID, o in casi particolari dalle sue altre versioni(PI, PD o P)



**Figura 12:** Controllo PID

Come possiamo osservare dalla figura 12 i controlli PID sono in retroazione e si basano su 3 componenti, proporzionale, integrale e derivata (infatti PID sta Proporzionale Integrale Derivata) il controllo può essere modificato attraverso la scelta dei tre coefficienti( $K_p$ ,  $K_i$ ,  $K_d$ ). Ovviamente queste tre componenti hanno caratteristiche differenti e compiti differenti infatti la componente proporzionale è direttamente proporzionale all'errore quindi maggiore sarà l'errore rispetto al setpoint, maggiore sarà il contributo di questa componente, l'integrale invece lavora attraverso l'accumulo nel tempo dell'errore, e può essere utile nei casi di sistemi non bilanciati, infine la derivata lavora con la velocità di variazione dell'errore.

La derivata è la componente più complessa da utilizzare(infatti a volte si preferisce non utilizzarla, optando per un controllore di tipo PI) soprattutto con segnali molto rumorosi la derivata porta ad un'uscita totalmente inutilizzabile. Nel caso di un drone, la componente derivata è molto importante ,in quanto in fase di stabilizzazione riesce a compensare i rapidi movimenti del drone,però come abbiamo detto prima tale componente è molto soggetta ai segnali rumorosi come la nostra IMU. Proprio per questo motivo si è deciso di filtrare ulteriormente l'uscita della nostra derivata attraverso un high-pass filter.

### 4.2.1 Spiegazione schema di controllo del drone

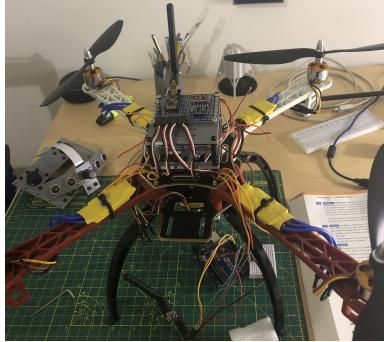
Il drone si basa su 7 controller PID, divisi su tre livelli principali. Il primo livello è quello che gestisce direttamente la posizione, questo livello utilizza solamente la componente proporzionale visto che serve solo che lavori proporzionalmente sull'errore tra posizione reale e posizione desiderata. Il secondo livello è utilizzato solamente per la stabilizzazione dei due assi orizzontali, utilizza come set-point l'uscita dei controller della posizione e lavora solamente con gli angoli. Il terzo ed ultimo livello è quello deputato a gestire le velocità angolari, utilizza come set-point l'uscita dal controller dell'angolo e il valore di uscita sarà direttamente utilizzato per la gestione dei motori. I valori di uscita dai vari controller vanno utiliz-

zati per controllare i motori, di tale compito si occupa il blocco Motor Calc nella figura 11, il quale in input prende tutte le uscite dei vari controller e in output fornisce i comandi pwm per i motori.

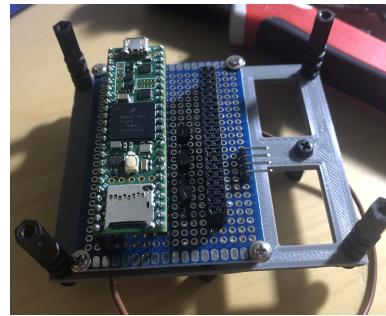
$$\begin{aligned} M_{front-left} &= t + U_{rateX} - U_{rateY} + U_{alt} \\ M_{back-left} &= t + U_{rateX} + U_{rateY} + U_{alt} \\ M_{front-right} &= t - U_{rateX} - U_{rateY} + U_{alt} \\ M_{back-right} &= t - U_{rateX} + U_{rateY} + U_{alt} \end{aligned}$$

Dove  $M_{front-left}$ ,  $M_{back-left}$ ,  $M_{front-right}$  e  $M_{back-right}$  sono i comandi per motori,  $t$  è la rotazione base dei motori calcolata in modo tale da generare la forza necessaria per mantenere il drone perfettamente in hovering (valore calcolato sperimentalmente),  $U_{rateX}$  è l'uscita del controllore per l'asse X del rotation rate,  $U_{rateY}$  è l'uscita del controllore per l'asse Y del rotation rate e infine  $U_{alt}$  è l'uscita del controller che gestisce l'altezza.

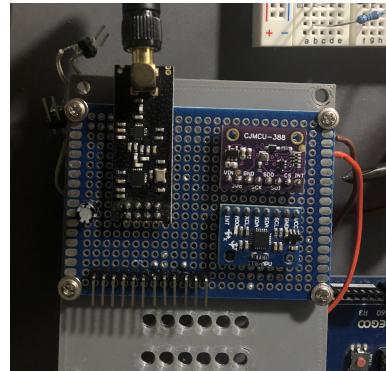
## 5 Drone completato



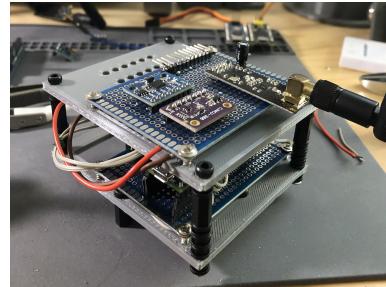
**Figura 13:** Drone completo



**Figura 14:** Flight computer drone



**Figura 15:** Modulo dei sensori del drone



**Figura 16:** Computer di volo assemblato