UNIVERSITÀ
POLITECNICA
DELLE MARCHE

FACOLTA' DI INGEGNERIA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Programmazione di un microcontrollore di volo per droni quadrirotore

Flight controller Software Development for a quadcopter

Relatore:

Professor Andrea Bonci

Candidato:

Ebram Ebrahim Adib Rezkalla

Anno Accademico 2021-2022

INDICE

Obiettivo

La presente tesi di laurea triennale ha l'obiettivo di progettare e controllare un prototipo di un drone quadrirotore. Nei prossimi paragrafi si esamineranno le caratteristiche e le componenti tecniche dei quadrirotori. Verrà altresì esplicato l'assemblaggio di un quadrirotore, tenendo conto di tutte le potenzialità dei suoi componenti ad esso applicati e di come gestire la progettazione del software del flight controller che, grazie all'uso di un microcontrollore della scheda STM32, è possibile realizzarlo. Infine, saranno illustrate le prove effettuate sul quadrirotore esaminandone risultati, spiegando poi le tecniche risolutive per compensare i vari errori.

1 QUADRIROTORI

1.1 INTRODUZIONE

I quadrirotori, una delle tecnologie più avanzate ed emergenti dell'era moderna, sono una forma di aeromobile a quattro eliche che possono essere utilizzati per una varietà di scopi, come la fotografia aerea, la ricerca scientifica, la sorveglianza e persino per hobby. Sono dotati di quattro rotori e di una serie di sensori e sistemi di controllo ,flight controller, che consentono loro di volare in modo stabile e controllato.

Il flight controller è un dispositivo elettronico che viene utilizzato per il controllo del volo di un veicolo aereo. Si tratta di un sistema di controllo che permette di monitorare e regolare il comportamento del veicolo in fase di volo. Il flight controller viene utilizzato sia per i velivoli commerciali sia per quelli militari e può essere considerato come il "cervello" del veicolo aereo. La sua funzione principale è quella di garantire il mantenimento del veicolo in volo, rendendolo stabile e sicuro; per fare ciò, il flight controller, dev'essere in grado di monitorare continuamente le condizioni del veicolo e dell'ambiente circostante e di intervenire, in caso di necessità, per mantenere il velivolo sulla rotta prestabilita.

1.2 CONFIGURAZIONE E CARATERISTICHE DI VOLO

1.2.1 CONFIGURAZIONE

Le configurazioni tipiche di un quadrirotore sono due e vengono definite dalla direzione di volo in relazione alla posizione dei motori; nella figura seguente si osserva la configurazione (X) e quella (+) [4]:

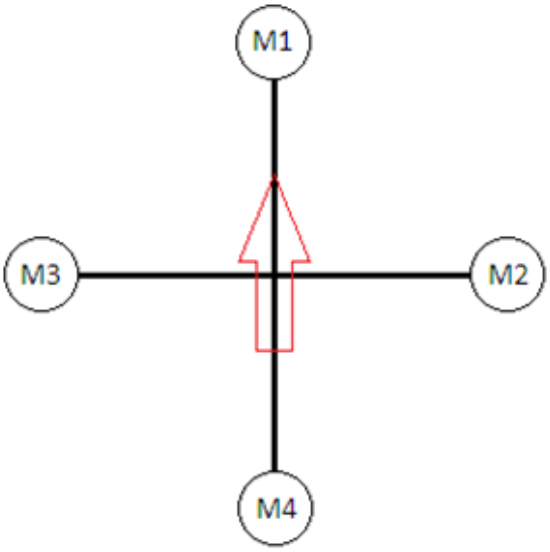


Fig 1.1: Configurazione +

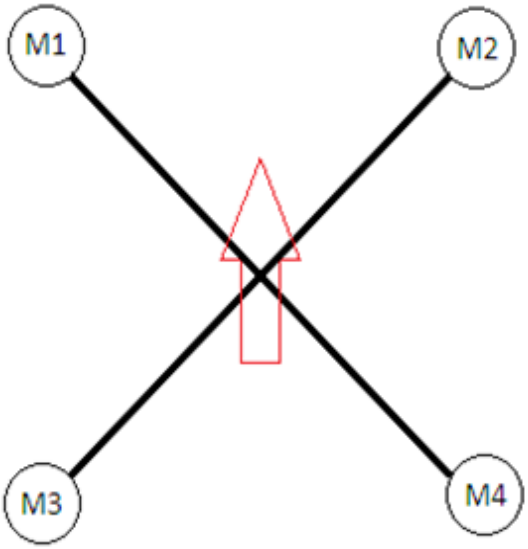


Fig 1.2: ConfigurazioneX

La freccia rossa indica il senso di marcia in avanti. Nella configurazione + la direzione di volo corrisponde all'asse creato dai motori M1 e M3 mentre, nella configurazione X la direzione di volo corrisponde all'asse creato tra i motori M1 e M4 e l'asse creato tra i motori M2 e M3.Per lo sviluppo del software è stata scelta la configurazione +.

1.2.2 CARATERISTICHE DI VOLO

IL quadrirotore è caratterizzato da tre movimenti principali Pitch,Roll e Yaw che di seguito vengono spiegati in dettaglio [5].

PITCH

Considerando che il quadrirotore in volo è perfettamente stabile (situazione teorica, Per il movimento in avanti il flight controller montato al centro del quadrirotore, dovrà aumentare la velocità di rotazione del motore posteriore M3 e diminuire la velocità di rotazione del motore anteriori M1. In questo caso il quadrirotore si inclinerà in avanti permettendone il movimento verso quella direzione.

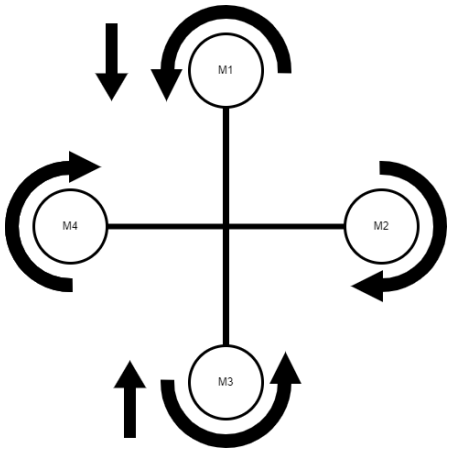


Fig 1.3 Movimento Pitch in avanti

Per portare il quadrirotore indietro il flight contoroller dovrà eseguire l'operazione opposta, ovvero aumentare la velocità di rotazione del motore M1 e diminuire la velocità di rotazione del motore M3.

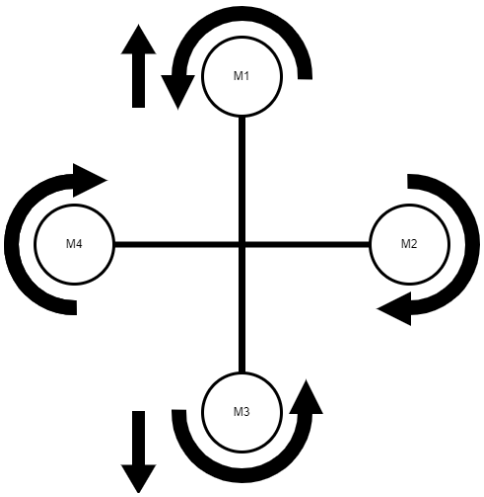


Fig 1.4 Movimento Pitch in dietro

Roll

Il movimento Roll è responsabile dello spostamento laterale del quadrirotore. Anche in questo caso il movimento avviene variando la velocità dei motori in modo tale da poter inclinare su un lato il quadrirotore permettendone quindi lo spostamento a destra o sinistra.

Spostamento a Destra (Relativo all'osservatore posto di fronte al quadrirotore) il motore M4 aumenta la velocità di rotazione, il motore M2 diminuisce la velocità di rotazione. Si ottiene una inclinazione a destra con conseguente movimento del velivolo verso quella direzione.

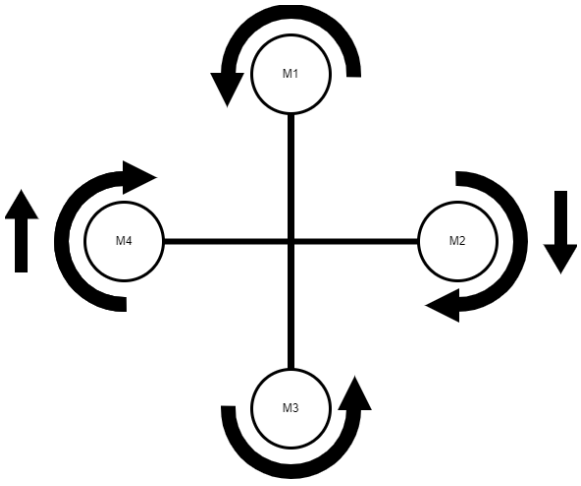


Fig 1.5 Movimento Roll a destra

Spostamento a Sinistra (Relativo all'osservatore posto di fronte al quadrirotore) il motore M2 aumenta la velocità di rotazione, il motore M4 diminuisce la velocità di rotazione. Si ottiene una inclinazione a sinistra con conseguente movimento del quadrirotore verso quella direzione.

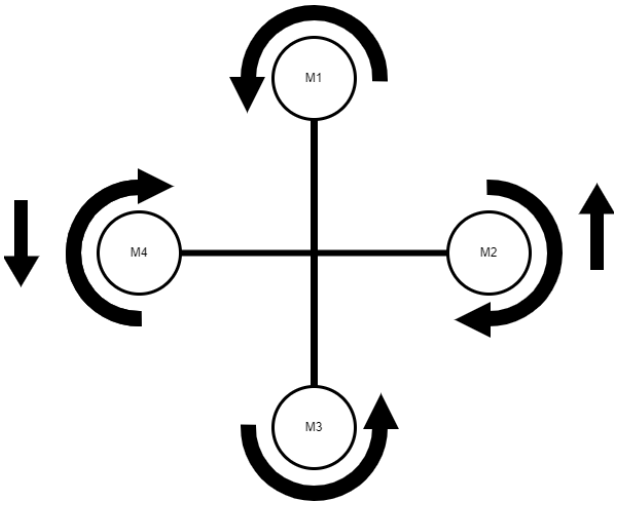


Fig 1.6 Movimento Roll a sinistra

Yaw

L'altro movimento che può eseguire il quadrirotore è la rotazione su sé stesso, definita Yaw. Il movimento avviene variando la velocità di rotazione di due motori posizionati nello stesso asse.

Rotazione oraria effettuata aumentando la velocità di rotazione dei motori M2 ed M4 e diminuzione della velocità di rotazione dei motori M1 e M3.

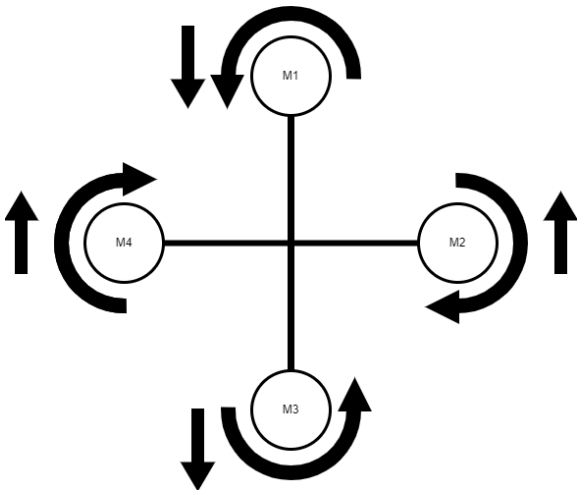


Fig 1.7 Movimento Yaw orario

Rotazione antioraria CCW effettuata diminuendo la velocità di rotazione dei motori M2 ed M4 e aumentando la velocità di rotazione dei motori M1 e M3.

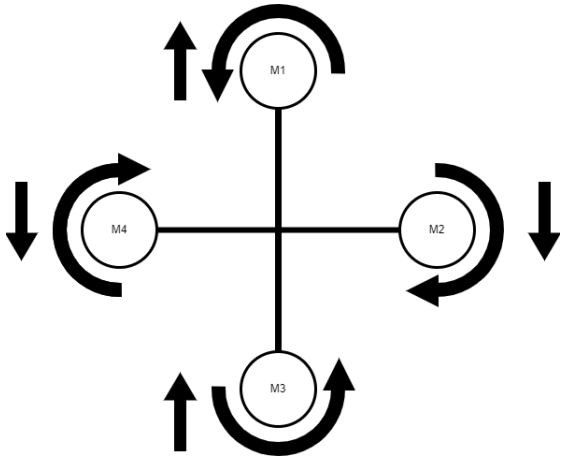


Fig 1.8 Movimento Yaw antiorario

2 SCELTA HARDWARE

In questo capitolo si presentano tutti i componenti meccanici e sensori che sono stati utilizzati, descrivendone le prestazioni. Inoltre, si indica la scelta corretta di questi componenti ai fini della compatibilità.

2.1 SCHEDA STM32H745ZI_Q NUCLEO-144

Per lo sviluppo del software del FC è stata utilizzata la scheda Nucleo-144, questa mette a disposizione due microcontrollori: Arm Cortex-M7 e Arm Cortex-M4. Durante l'implementazione è stato programmato solamente il microcontrollore Arm_Cortex-M4, con 240MHz e una flash memory molto performante e veloce che acquisisce i dati dai sensori e li seleziona utilizzando il filtro Kalman, per poi mandare i vari segnali PWM ai motori.

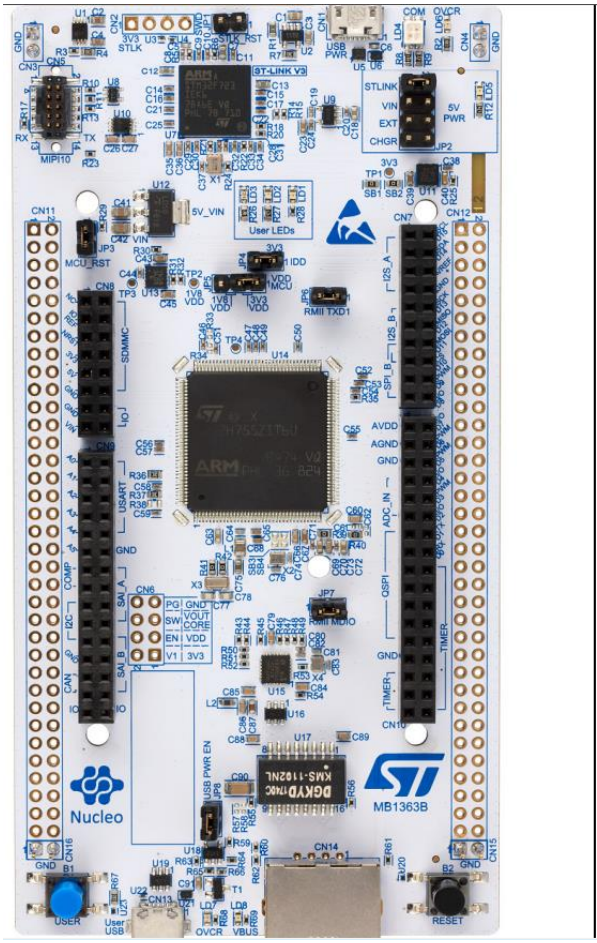


Fig 2.1: Vista fronte scheda

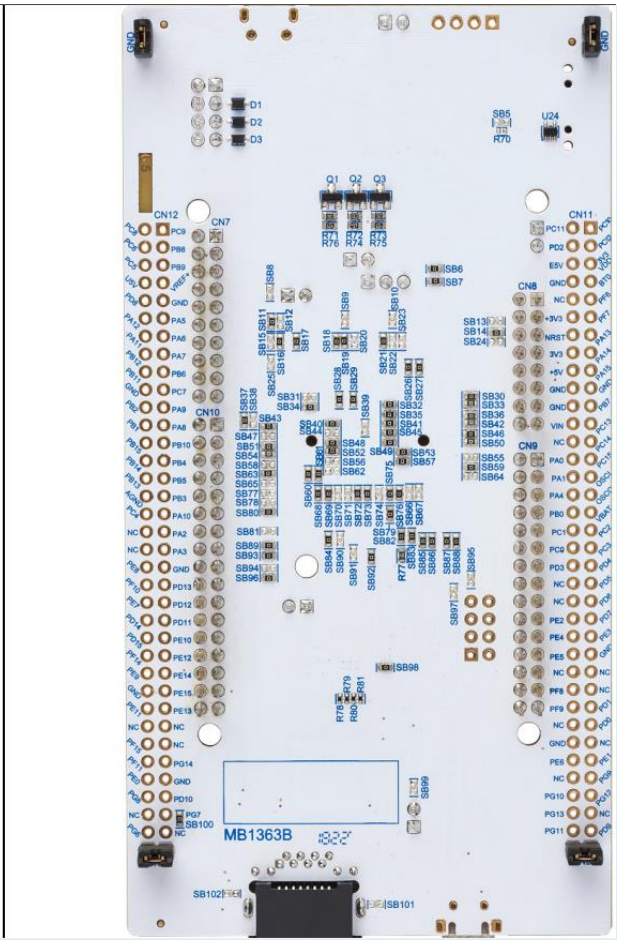


Fig 2.2: Vista retro-scheda

Tabella delle caratteristiche fondamentali per lo sviluppo del FC [11]:

Caratteristica	Nucleo-144
Voltaggio di ingresso	5-12V
Canali PWM	14 canali (per il quadrirotore servono solo 8 canali di PWM,4 canali per i motori e altri 4 canali che vengono usati in modalita' input capture per leggere i segnali dal telecomando)
GPIO (general-purpose input output)	16 I/O
I2C	Periferica I2C serve per la comunicazione con i sensori
Flash memory	2 Mbytes
Clock speed	240MHz

Tabella 2.1: caratteristiche scheda STM32

La configurazione delle periferiche della scheda STM32H745ZI_Q avviene utilizzando la piattaforma “STM32CubeIDE”, messa a disposizione da parte dell’azienda produttrice che sarà poi approfondita nel capitolo 5. [18]

2.2 TELAIO

Per la scelta del telaio è sempre meglio prediligerne uno leggero, resistente ed il più robusto possibile; basandosi su questi criteri si è deciso di acquistare “DJI F450”, il telaio più impiegato nei quadrirotori di sperimentazione. Nel telaio è integrata una scheda di distribuzione per collegare i motori alla batteria [12].

Fig 2.3: Telaio DJI F450

Fig 2.4: Scheda di distribuzione

Materiale	Plastica
Lunghezza asse	450 mm
Peso	190 grammi

Tabella 2.2: caratteristiche Telaio

2.3 MOTORI

Sui quadrirotori, solitamente, vengono impiegate due tipologie di motori elettrici: il motore di tipo brushed e di tipo brushless. I motori brushless sono principalmente utilizzati nei quadrirotori professionali, mentre i motori brushed, invece, vengono impiegati nella realizzazione di oggetti di semplice utilizzo, come ad esempio i giocattoli. In questo caso, per il quadrirotore, sono stati applicati dei motori di tipo brushless.

I motori brushless si controllano tramite un segnale PWM di frequenza 50Hz. Per avviare questo tipo di motori, prima devono essere armati. L'armamento è una procedura di sicurezza che consiste nell'inviare un certo segnale PWM con un certo duty cycle. Questi spetti di controllo saranno approfonditi nel capitolo 5.

È di fondamentale rilevanza scegliere dei motori brushless affidabili, dunque qualitativamente migliori, per assicurarsi un'alta efficienza prestazionale, che comporta quindi una risposta più rapida nel controllo del quadrirotore.

Una buona efficienza di questo tipo di motore, non solo ci consente di ottenere un quadrirotore di alto livello, bensì ci assicura un margine di sicurezza maggiore. Basti pesare alla pericolosità che può potenzialmente avere un quadrirotore se, durante il volo, si guastassero uno o più motori brushless, compromettendo la stabilità del drone e quindi mettendo a rischio chi e cosa c'è alla sua portata.

Inoltre, è indispensabile che i motori siano abbastanza potenti da poter sollevare il quadrirotore e portarlo ad eseguire vari movimenti. Infine, ci tengo a precisare che i motori sono quasi privi di vibrazioni, poiché qualsiasi vibrazione causerebbe dei rumori nell' unità di misura inerziale(IMU).

Sulla base di questi criteri si è deciso di acquistare il motore brushless "DX2205" con 2300KV, progettato per aeroplani telecomandati e quadrirotori[13].

Fig 2.5: Motore Brushless DX2205

Item	KV(rpm/v)	Voltage(v)	Prop	Load Current(A)	Pull(g)	Power(w)	Efficioney (g/w)	Lipo	Weight(g) Approx
DX2205	2300	11.1	5045	19.2	660	213	3.1	2-4S	28
		14.8		27.6	950	408	2.3		

Fig 2.6: tabella caratteristiche motore DX2205 del produttore

Secondo le specifiche nella figura 2.6, ogni motore può dare una spinta fino a 950 grammi.

2.4 ESC

Un regolatore di velocità elettrico (ESC) è un circuito elettrico con lo scopo di variare la velocità del motore tramite un segnale PWM. Si cerca sempre un ESC che sia veloce e affidabile per le stesse ragioni esposte per i motori nel paragrafo 2.3. In base alla tensione di alimentazione del motore e le correnti che esso assorbe (vedi figura 2.6), si è deciso di acquistare un ESC con un peso di 10 grammi e con una tensione d'ingresso che varia dai 2 ai 4S (con S si intende cella, ogni cella è di 3.7 Volt) e che supporta fino a 40 Ampere. È sempre preferibile avere un margine di corrente del 40% in più rispetto alle correnti che assorbe il motore.

Fig 2.7: Esc per motori brushless

2.5 ELICHE

I criteri di scelta delle eliche sono meno restrittivi rispetto a quelli dei motori. Sono sempre preferibili le eliche leggere, resistenti agli urti esigui, e allo stesso modo in grado di fornire una spinta sufficiente a far volare agevolmente il quadrirotore.

La scelta delle eliche dipende anche dal momento di coppia che riesce a sviluppare il motore. In questo caso, secondo la tabella delle caratteristiche del motore in figura 4, si possono solamente applicare le eliche 5045 a 3 pale con un diametro di 5 pollici e con un peso di 5 grammi ciascuna [14].

Fig 2.8: eliche 5045

2.6 UNITÀ DI MISURA INERZIALE (IMU)

IMU è un sensore che consente di misurare l'accelerazione e la velocità angolare, ogni misura viene fatta su un sistema a 3 assi, quindi risulteranno 3 dati per l'accelerazione (Ax,Ay,Az) e 3 dati per la velocità angolare (Gx,Gy,Gz). Perciò, nel complesso, si avranno 6 variabili che esprimono la posizione nello spazio del nostro sensore; ed è proprio grazie a queste variabili che si può stabilire se un oggetto è in movimento, se sta traslando, ruotando o se è inclinato rispetto al pavimento.

Per lo sviluppo del FC si è deciso di acquistare MPU-6050, che ha a disposizione sia il giroscopio, per misurare le velocità angolari, che l'accelerometro, per la valutazione quantitativa delle varie accelerazioni. Questo sensore è molto preciso, in quanto contiene un convertitore da analogico a digitale 16-bit per ciascun canale [15].

La trasmissione dei dati tra il microcontrollore e il sensore MPU-6050 avviene tramite il protocollo di comunicazione I2C.

I2C è un protocollo di comunicazione seriale sincrono che permette a molti dispositivi di scambiarsi dati fra loro utilizzando solo due cavi, uno per i dati ed uno per sincronizzare la comunicazione [12].

Il bus I2C è composto in totale da 4 cavi:

Linea	Descrizione
Vdd	Tensione di riferimento
SCL	Clock per la sincronizzazione della comunicazione
SDA	Linea su cui transitano i dati
GND	Massa di riferimento

Fig 2.9: MPU-6050

2.7 BATTERIA

I motori e i sensori del quadrirotore sono tutti alimentati tramite una batteria di tipo lithium polymer battery (LiPo) con una tensione nominale (o media) di 3,7 V/cella e una tensione massima di 4,2 V/cella.

La batteria può contenere fino a 6 celle. Una volta che la cella risulta completamente carica, fornirà 4,2 V e, come limite inferiore di scarica, 3,5 V. Può essere fattore di rischio far scaricare la batteria fino al raggiungimento di un

valore inferiore a 3.5 V, questo perché la resistenza della batteria aumenta, causandone il riscaldamento ed il rigonfiamento.

La scelta della batteria dev'essere ben ponderata in base a molti fattori, tra i più rilevanti c'è il peso, in quanto la batteria è la componente che di solito pesa di più nel quadrirotore. Se si decide di utilizzare una batteria che fornisce maggiore afflusso di corrente, bisogna tener presente che il suo peso sarà maggiore, a discapito quindi dei motori, e questo può di conseguenza ridurre notevolmente il tempo di volo; al contrario, una batteria che fornisce meno corrente peserà meno, ma allo stesso tempo può non essere in grado di dare la potenza necessaria a far decollare il quadrirotore. Un altro fattore da non sottovalutare, nella scelta di una batteria, è la tensione di funzionamento dei motori che, nel nostro caso, può essere 11.1 V o 14.8 V (Vedi figura 2.6), anch'essa scelta prendendo in considerazione il peso che vado ad apportare.

In base a questi criteri ho scelto la batteria "OVONIC Air" [16] :

Fig 2.10: Batteria LiPo

Voltaggio e Numero di celle	3S1P (tre celle collegate in serie e una cella collegata in parallelo, quindi 3 celle *3.7 V = 11.1V)
Capacità	3000mAh
Capacità di Scarica	50 C
Peso	196 grammi
Tipo di connettore	T stile Dean

Tabella 2.2: caratteristiche della batteria

La capacità della batteria è espressa in mAh o Ah e può essere utilizzata per stimare il tempo di volo. La capacità della batteria è definita, più specificamente, come il numero di ore di corrente (o potenza) che la batteria può fornire. Le unità di misura sono l'ampere-ora (Ah) e il watt-ora (Wh). Se una batteria ha una capacità di 1 Ah, può assorbire 1 A di corrente per un'ora; allo stesso modo, se la capacità è di 1 Wh, la batteria fornirà 1 W di potenza per un'ora.

La tensione nominale della batteria consente di determinare la velocità del motore. Poiché le velocità dei motori sono definite in Kv con l'unità RPM/Volt, il numero di volt che la batteria può fornire determina la velocità di rotazione del motore. Nel nostro caso, la tensione è di 11.1 V che moltiplica 2300KV del motore, perciò si avranno 25530 giri al minuto (RPM).

Il fattore di scarica, o fattore C, determina la velocità con cui la batteria può scaricarsi in sicurezza, permettendo di valutarne il tempo di scaricamento. Questo fattore aiuta a valutare la corrente massima che si può assorbire dalla batteria. In questo caso, la batteria ha un fattore C di 50 e una capacità di 3000 mAh/ 3 Ah, quindi la corrente massima che si può assorbire in sicurezza dalla batteria è di 50 * 3 = 150 Ah, che dev'essere sempre maggiore alla corrente massima assorbita dai quattro motori, 4*20A=80A.

CARICABATTERIE: bisogna acquistare un caricabatterie adatto basandosi sul numero di celle della propria batteria, dunque, si è acquistato un caricatore per le batterie a 3 celle.

Fig 2.11: Caricabatterie per batterie a 3 celle.

2.8 TELECOMANDO

Il telecomando è essenziale per il controllo del quadrirotore, ecco perché bisogna munirsene al fine di fare le prime prove di accertamento del corretto funzionamento di tutti i componenti e per poter portare avanti il progetto, fino al raggiungimento dello scopo finale: un volo stabile del quadrirotore, sotto tutti i punti di vista, e con la giusta reattività agli stimoli.

Il telecomando deve avere un range di controllo di almeno 100 metri e almeno 4 canali per riuscire a controllare la spinta dei motori (l'altezza del quadrirotore) e i tre movimenti principali: pitch, roll e yaw .

Per telecomandare il quadrirotore si è deciso di acquistare il telecomando "MicroZone MC6C"[17]:

Fig 2.12: trasmettitore con ricevitore.

TABELLA DELLE CARATTERISTICHE DEL TELECOMANDO UTILIZZATO

Numero dei Canali	6 canali
Range di controllo	800 metri
Tensione di funzionamento del ricevitore	4,5--8,5 V
Corrente massima del ricevitore	120 mA
Tipologia segnale in uscita dal ricevitore	6 segnali PWM e 1 segnale SBUS
Peso del ricevitore	7 grammi

Tabella 2.3: caratteristiche del ricevitore radiocomandato

Il ricevitore del telecomando,che sarà installato a bordo del quadrirotore, invia in uscita su ogni canale, un segnale PWM di frequenza 50Hz.

Siccome il ricevitore mette a disposizione sei canali di controllo; quindi, in uscita si ha sei segnali PWM. Per il controllo del quadrirotore servono solo quattro canali di controllo: spinta, Ptich, Roll e Yaw.

Nel capitolo 5.5 verrà spiegato il metodo per l'acquisizione di questi quattro segnali PWM.

2.9 UBEC (Ultimate Battery Eliminator Circuit)

UBEC è un componente che consente di abbassare la tensione fornita dalla batteria per alimentare gli altri costituenti del quadrirotore, come la scheda STM32, il sensore IMU e il ricevitore, cha hanno una tensione di funzionamento di 5 V.

Si è acquistato un UBEC che ha una tensione di ingresso compresa tra i 2 e i 6 S, mentre in uscita fornisce una tensione di 5-6 V, regolabile tramite un jumper a bordo. Si aggiunge che, questo UBEC, è in grado di supportare fino a 5A.

Fig 2.13: ULTIMATE Battery Eliminator Circuit (UBEC)

2.10 CALCOLO SPINTA MOTORI

In base alle proprietà dei componenti selezionati ed esposti nei paragrafi precedenti, si è già in grado di calcolare la spinta massima dei quattro motori, prendendo in considerazione il peso totale del quadrirotore.

Basandosi sulla tabella delle caratteristiche dei motori (vedi figura 2.6), ogni motore, alla capacità del 100%, riesce a fornire una spinta massima di 660 grammi, avendo come eliche le “5045” (paragrafo 2.5) e alimentato da una batteria di 11.1 V (paragrafo 2.3).

Avendo quattro motori, la spinta massima totale sarà:

$$numero\ motori\ \times spinta\ unitaria = spinta\ totale$$

$$4 \times 660 = 2640\ grammi$$

Il peso complessivo del quadrirotore è di 880 grammi, quindi teoricamente il decollo avverrà a circa il 33% della capacità dei motori. Questo è più che sufficiente per ottenere un volo stabile e, in questo modo, il quadrirotore riuscirà ad esercitare dei movimenti più rapidi, che renderanno più semplice il controllo.

3 SCHEMA DI COLLEGAMENTI ED ASSEMBLAGGIO

In questo capitolo si presenta uno schema dei collegamenti dei vari costituenti e le fasi di assemblaggio del quadrirotore, descrivendone la configurazione. Inoltre, si espongono alcune tecniche di montaggio per ridurre al meglio le vibrazioni prodotte dai motori in fase di funzionamento.

3.1 SCHEMA DI COLLEGAMENTI

Di seguito lo schema dei collegamenti:

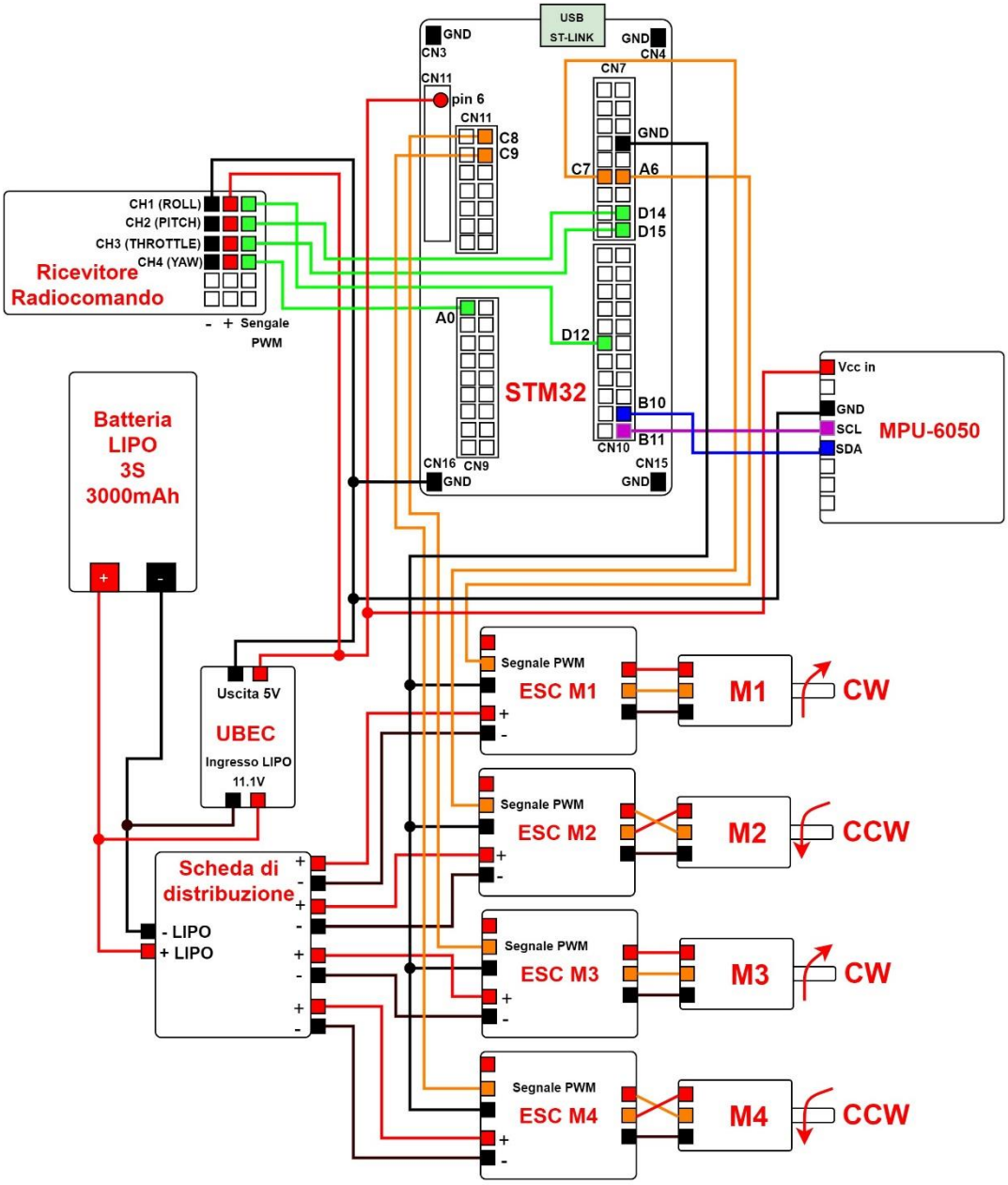


Fig 3.1 schema di collegamenti

3.2 ASSEMBLAGGIO QUADRIROTORE

L'assemblaggio del quadrirotore ha richiesto un po' di manualità, ma nel complesso, è risultato essere piuttosto semplice. I quattro bracci sono collegati ad una struttura centrale a cui è integrata la scheda di distribuzione, ove su più livelli

è montata: l'elettronica di controllo comprensiva dei sensori appositamente posizionati, la batteria LIPO e le quattro ESC per i motori. Alle estremità dei bracci sono stati installati i motori [12].

L'assemblaggio è quindi cominciato con la saldatura a stagno delle quattro ESC alla scheda di distribuzione:

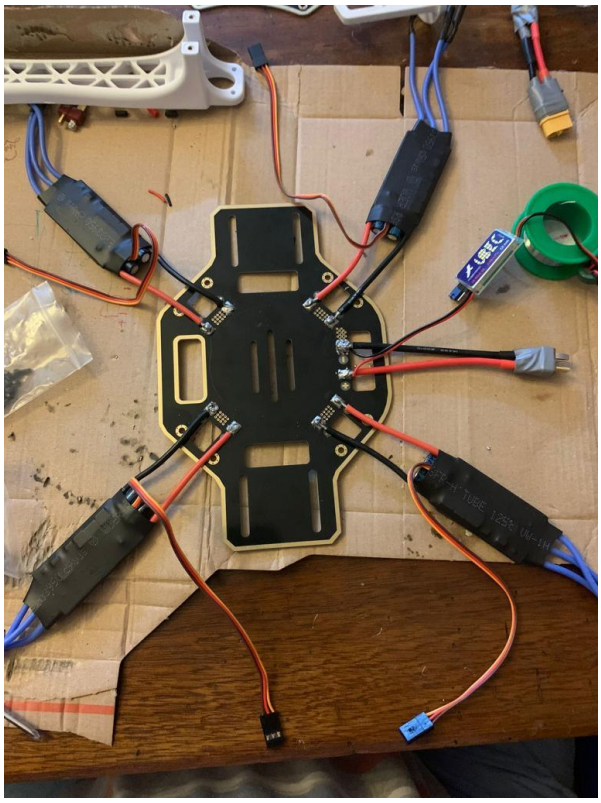


Fig.3.2: scheda di distribuzione con saldature completate

Si osservi che la scheda di distribuzione è semplicemente un circuito per distribuire alle quattro ESC la linea di alimentazione della batteria a 11.1V. In sostanza, questo PCB, non fa altro che mettere in comune i poli positivi e negativi sia delle quattro ESC che della batteria. Schematicamente ci si può riferire alla Fig. 3.1.

Dopo aver saldato le ESC, si è passati alla saldatura dei connettori a banana di diametro 3.5mm ai cavi di uscita dei motori.

Fig 3.3: motore senza connettori

Fig 3.4: motori con connettori

Dopo la saldatura dei connettori, si è passati ad installare i motori alle estremità dei bracci tramite l'uso di quattro viti per motore; questa fase è assai rilevante al fine di evitare eventuali vibrazioni in fase di funzionamento del quadrirotore, perciò è bene saper fissare adeguatamente i mortori. Sono state installate anche le ESC lungo i bracci tramite un nastro adesivo.

Fig 3.5: Motore installato sul braccio del quadrirotore

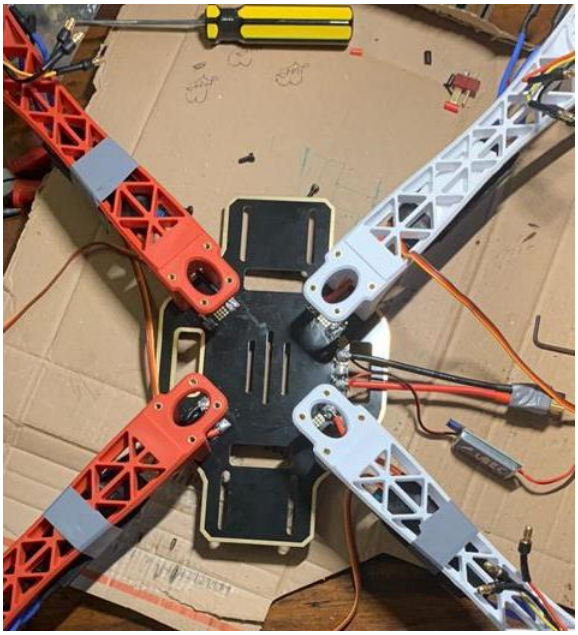


Fig 3.6: bracci montati la struttura centrale

Va sottolineato che il senso di rotazione specificato su ogni motore va rispettato, in quanto la filettatura presente sull'albero dello stesso deve avere un senso contrario al senso di rotazione. Non rispettando il senso di rotazione specificato, durante il funzionamento del motore, potrebbe svitarsi un dado (o più, dei quattro montati per fissare le eliche alla struttura) e conseguentemente smontare l'elica, a causa della forza dell'aria (attrito) che spinge su di essa in senso di svitamento.

L'inversione del senso di rotazione viene ottenuta invertendo uno dei due cavi del motore collegati all'ESC, schematicamente ci si può riferire alla Fig. 3.1.

Nel quadrirotore due motori sullo stesso asse girano nello stesso verso, come spiegato nei capitoli precedenti.

L'ultimo passaggio, prima di analizzare com'è stata installata l'elettronica di controllo e di trasmissione, è quello che concerne l'installazione delle eliche come in Fig.3.7.

Fig.3.7 motore con elica installata

Le eliche stesse hanno un verso di rotazione che deve coincidere con il verso di rotazione del motore su cui vengono montate. Sulla maggior parte delle eliche viene indicato il verso di rotazione, ma, nel caso non fosse indicato, si può comunque dedurlo osservando la forma dell'elica.

Il senso di rotazione è sempre verso il lato diretto non tagliente dell'elica, come si può notare nelle figure 3.8 e 3.9.

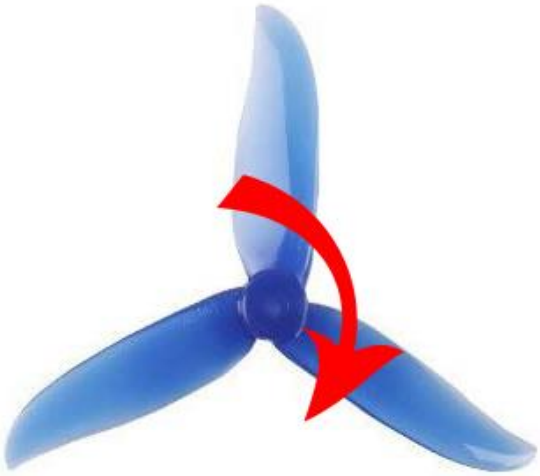


Fig.3.8 elica con verso rotazione orario

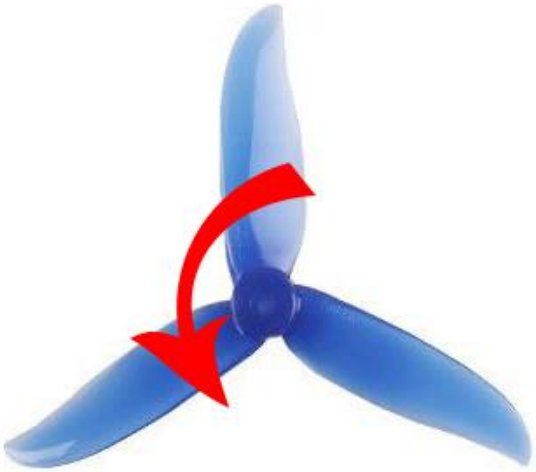


Fig.3.9 elica con verso di rotazione antiorario

Questo lo si può spiegare banalmente pensando a come l'aria viene gestita proprio dalla fisionomia dell'elica stessa. Essendo che il nostro fine è quello di far sollevare in volo il quadrirotore, bisogna produrre un flusso d'aria consistente verso il basso, in risposta all'attrazione gravitazionale, ecco perché, questa spinta opposta alla gravità, compenserà il peso della struttura, sollevandola da terra.

Al contrario, se le eliche girassero dal lato tagliente, il flusso d'aria verrebbe prodotto verso l'alto, risultando futile, nel nostro caso.

Infine, sono state installate le parti di elettronica di controllo, tra cui il sensore inerziale "MPU-6050".

Come primo passaggio sono stati saldati al sensore i cavi di alimentazione e di comunicazione come in Fig3.10. Di seguito, per avere una stima corretta e precisa delle velocità angolari e delle accelerazioni degli assi, il sensore è stato installato al centro di massa del quadrirotore.

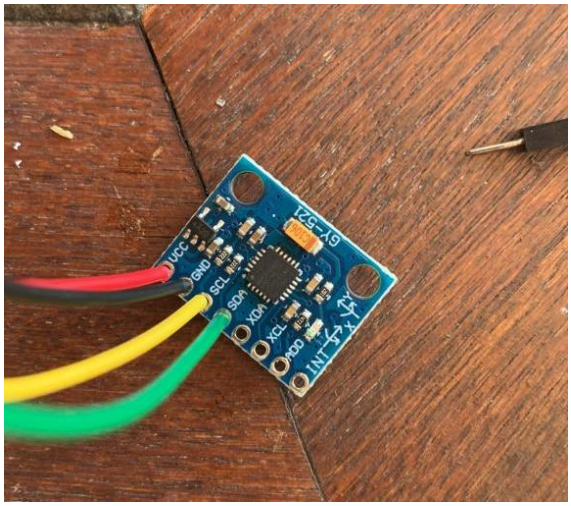


Fig.3.10 Sensore IMU con cavi saldati

L'installazione è avvenuta tramite un biadesivo fissato su un pezzo di gomma per filtrare meccanicamente le turbolenze prodotte durante il volo.

In base alla configurazione scelta (vedi paragrafo 1.2.1) il sensore deve essere orientato verso uno dei quattro motori in modo da essere allineato all'asse "X" o "Y".

Nell'ultima fase è stata montata la scheda STM32, il ricevitore del telecomando e infine la batteria al centro di massa, in quanto il bilanciamento del quadrirotore è essenziale per avere una buona stabilità in volo.

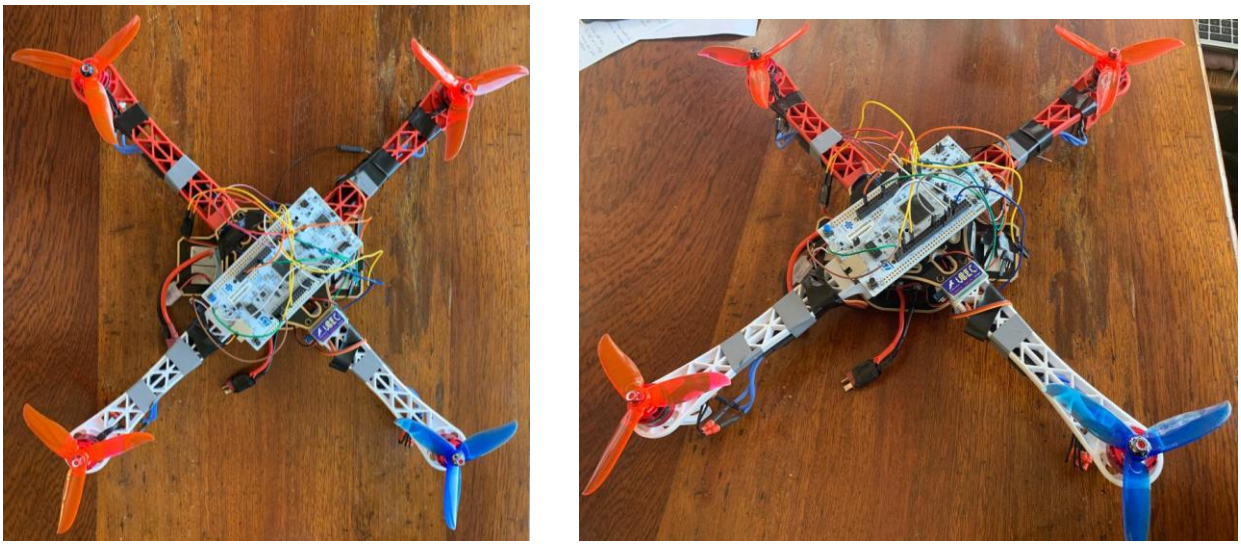


Fig.3.11 Quadrirotore completo

4 MODELLO MATEMATICO E CONTROLLO

In questo capitolo viene introdotto il modello matematico, che consiste praticamente nel modellare adeguatamente le dinamiche di un sistema. In questa fase vengono illustrate le tecniche e le equazioni usate per modellare un quadrirotore nel suo complesso e, dunque, vengono fissate le basi matematiche per la descrizione di un quadrirotore. Impostando il modello matematico, successivamente si possono ben definire le modalità di controllo del quadrirotore [1],[2].

4.1 MODELLO MATEMATICO DEL QUADRIROTORE

Il quadrirotore si muove in uno spazio tridimensionale, quindi è possibile definire un sistema di riferimento mobile solidale, con il baricentro del quadrirotore indicato con 'ABC' (Aircraft Body Center). La terna è composta dall'asse X, che punta verso uno dei motori, l'asse Z che invece punta in direzione terrestre ed, infine, l'asse Y, ortogonale alle altre due assi, come si vede in figura 4.1.

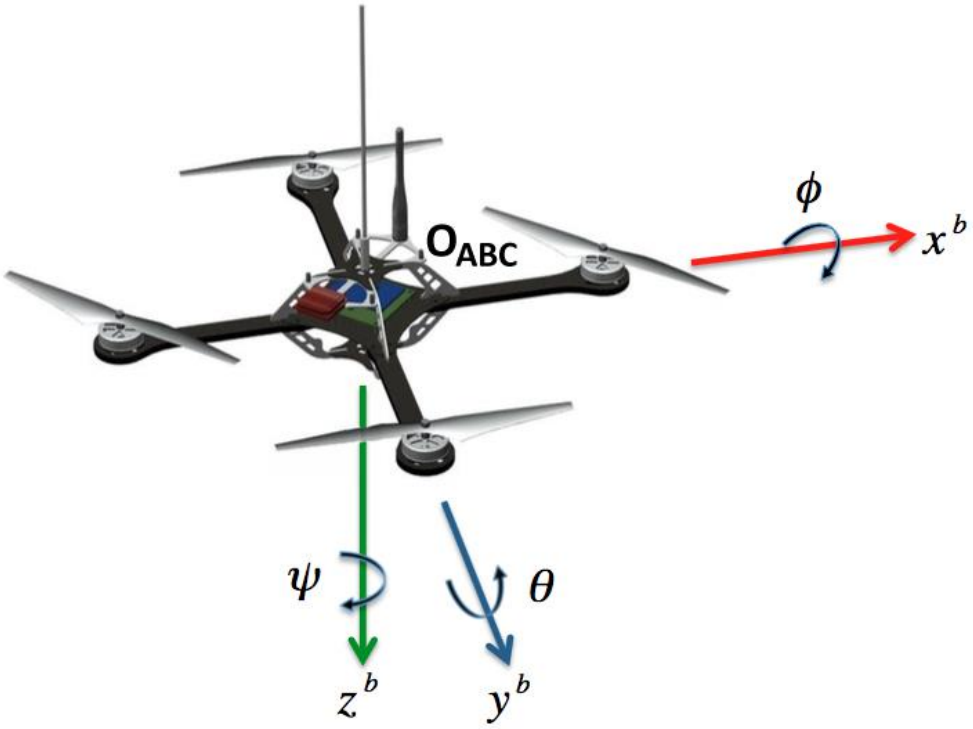


Fig.4.1 Sistema di riferimento ABC

Per poter procedere con il modello matematico, è necessario introdurre un sistema di riferimento fisso, rispetto al quale saranno espresse le dinamiche della terna mobile definita nel paragrafo precedente. Nello sviluppo di questo modello è stato scelto di utilizzare, come sistema di riferimento fisso, una terna chiamata NED (North-East-Down), dove l'asse x punta verso il nord, l'asse y verso est, mentre il l'asse z punta verso il basso alla superficie terrestre.

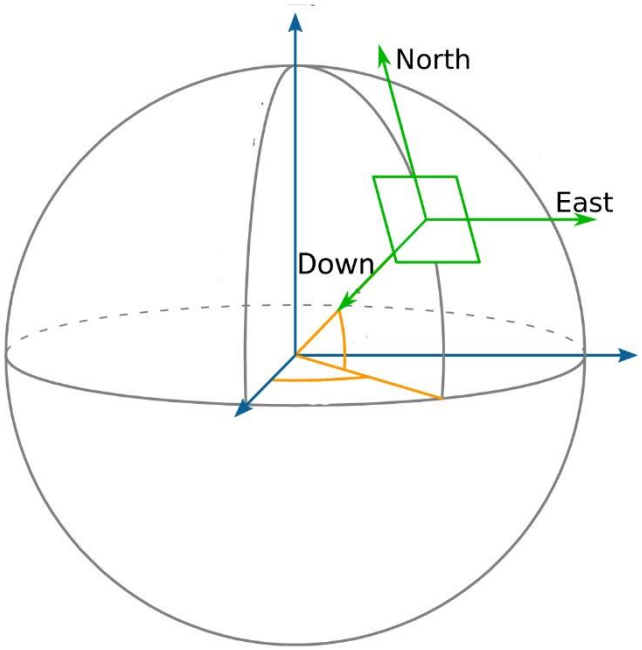


Fig4.2. Sistema di riferimento fisso NED

L'origine del NED è generalmente definita come il punto di partenza o di decollo del quadrirotore. La posizione del quadrirotore è definita come la posizione dell'origine O_{ABC} rispetto all'origine O_{NED}. L'immagine che segue mette in luce le due terne ed i rispettivi versori che le formano.

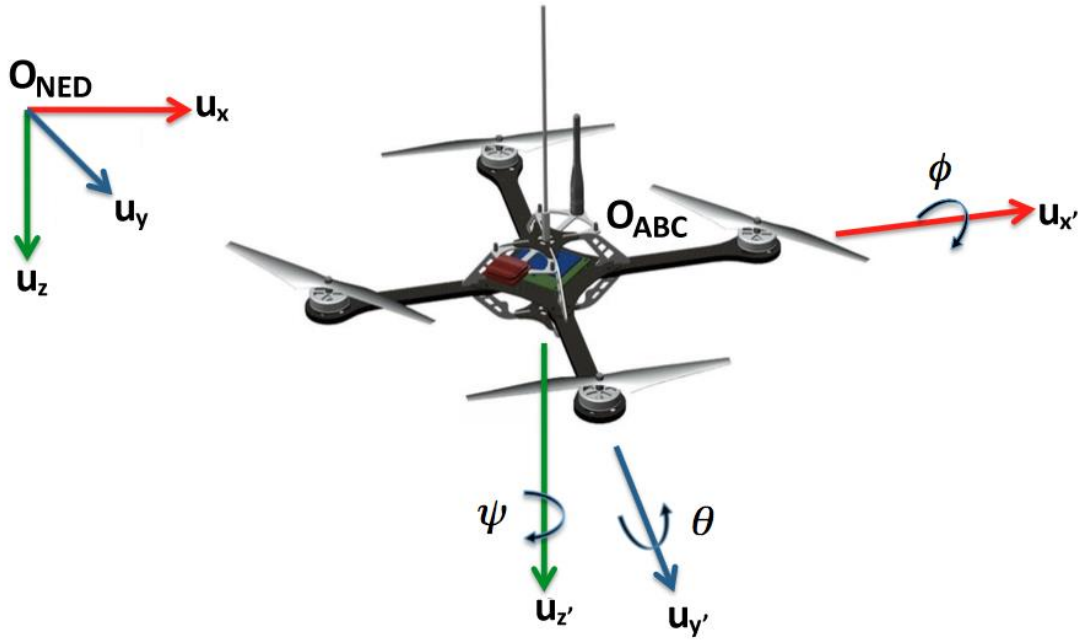


Fig.4.3: i sistemi di riferimento NED e ABC.

Le dinamiche del sistema sono espresse attraverso lo stato di esso che, nel caso del quadrirotore, corrispondono a dodici equazioni, le quali ne descrivono l'assetto e la posizione a sei gradi di libertà. Le variabili di interesse sono quindi sei per il sistema NED ed altre sei per il sistema ABC. Nello specifico, definiamo come V_B e ω_B il vettore di velocità lineari e angolari del velivolo nella terna O_{ABC} con le seguenti variabili di stato:

$$V_B = [u \quad v \quad w]^T \qquad \omega_B = [p \quad q \quad r]^T \qquad (4.1)$$

Per il sistema NED invece consideriamo le posizioni lineari:

$$\Gamma_{NED} = [x \quad y \quad z]^T \qquad (4.2)$$

e gli angoli di roll pitch e yaw

$$\Theta_{NED} = [\phi \quad \theta \quad \psi]^T \qquad (4.3)$$

Per trasformare i vettori tra diversi sistemi di riferimento, si utilizzano le matrici di rotazione. Una matrice di rotazione **R** che trasforma una rappresentazione vettoriale dalla terna **b** alla terna **a**, è indicata con $\boldsymbol{R_b^a}$. Generalmente una matrice di rotazione **R** è una matrice che soddisfa la seguente formula:

$$RR^T = R^T R = I \qquad (4.4)$$

Ciò implica che R è ortogonale e, di conseguenza, l'inversa della matrice di rotazione è data da $\boldsymbol{R^{-1} = R^T}$, quindi:

$$R_b^a = (R_b^a)^{-1} = (R_b^a)^T \qquad (4.5)$$

Si procede utilizzando la convenzione di Yaw, Pitch e Roll (corrispondente a zyx), indicate rispettivamente con **ψ**, **θ**, **ϕ**. Dunque, la prima matrice di rotazione corrisponde a una rotazione attorno all'asse Z (Yaw), la seconda attorno all'asse Y (Pitch)e l'ultima attorno l'asse X (Roll).

$$R_{z,\psi} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.6)$$

$$R_{y,\theta} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \qquad (4.7)$$

$$R_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \qquad (4.8)$$

Di conseguenza, si ottiene la matrice di rotazione da ABC a NED come segue:

$$R_{(\psi,\theta,\phi)} = R_{z,\psi} R_{y,\theta} R_{x,\phi} \qquad (4.9)$$

$$R = \begin{bmatrix} \mathrm{c}(\psi)\,\mathrm{c}(\theta) & \mathrm{c}(\psi)\,\mathrm{s}(\theta)\,\mathrm{s}(\phi) - \mathrm{s}(\psi)\,\mathrm{c}(\phi) & \mathrm{c}(\psi)\,\mathrm{s}(\theta)\,\mathrm{c}(\phi) + \mathrm{s}(\psi)\,\mathrm{s}(\phi) \\ \mathrm{s}(\psi)\,\mathrm{c}(\theta) & \mathrm{s}(\psi)\,\mathrm{s}(\theta)\,\mathrm{s}(\phi) + \mathrm{c}(\psi)\,\mathrm{c}(\phi) & \mathrm{s}(\psi)\,\mathrm{s}(\theta)\,\mathrm{c}(\phi) - \mathrm{c}(\psi)\,\mathrm{s}(\phi) \\ -\mathrm{s}(\theta) & \mathrm{c}(\theta)\,\mathrm{s}(\phi) & \mathrm{c}(\phi)\,\mathrm{c}(\phi) \end{bmatrix} \qquad (4.10)$$

Dove c(∗) = cos(∗), s(∗) = sin(∗) e t(∗) = tan(∗).

La relazione tra il vettore di velocità lineare V_B nel sistema ABC e il vettore di velocità $\dot{\Gamma}$ nel sistema fisso terrestre NED è quanto segue:

$$\dot{\Gamma}_{NED} = R.V_B \qquad (4.11)$$

Invece, il vettore di velocità angolare ω_B del sistema ABC è legato al vettore di velocità angolare $\dot{\Theta}$ del sistema NED, tramite una matrice di trasformazione T.

$$\dot{\Theta}_{NED} = \mathrm{T}.\,\omega_B \qquad (4.12)$$

Dove

$$T = \begin{bmatrix} 1 & \mathrm{s}(\phi)\,\mathrm{t}(\theta) & \mathrm{c}(\phi)\,\mathrm{t}(\theta) \\ 0 & \cos(\phi) & -\mathrm{s}(\phi) \\ 0 & \mathrm{s}(\phi)\,/\,\mathrm{c}(\theta) & \mathrm{c}(\phi)\,/\,\mathrm{c}(\theta) \end{bmatrix} \qquad (4.13)$$

Arrivati a questo punto è possibile applicare la seconda legge di Newton formulando due ipotesi:

• L'origine della terna ABC corrisponde al CDM del quadrirotore;

• Gli assi della terna ABC corrispondono con gli assi principali di inerzia (questo ci permette di avere un tensore di inerzia diagonale e di annullare i prodotti centrifughi derivanti da un tensore di inerzia non diagonale).

$$m.\ddot{\Gamma}_{NED} = \sum_{i=1}^{n} F_i = F_{NED} \qquad (4.14)$$

In termini angolari:

$$I.\ddot{\Theta}_{NED} = \sum_{i=1}^{n} \tau_i = \tau_{NED} \qquad (4.15)$$

A questo punto, sostituendoci l'equazione 4.8, si può scrivere:

$$m.\dot{\widehat{RV}}_B = RF_B \qquad (4.16)$$

Quindi, facendo la derivata del prodotto:

$$m(R\dot{V}_B + \dot{R}V_B) = RF_B \qquad (4.17)$$

Ricordandosi che la velocità di una terna si può esprimere come velocità (lineare) della terna stessa più la sua rotazione, si ha:

$$mR(\dot{V}_B + \omega_B \times V_B) = RF_B \qquad (4.18)$$

Ovvero

$$m(\dot{V}_B + \omega_B \times V_B) = F_B \qquad (4.19)$$

Mentre per i termini angolari possiamo scrivere similmente a

prima:

$$I.\dot{\widehat{T\omega_B}} = T\tau_B \qquad (4.20)$$

$$I.T(\dot{\omega}_B + \omega_B \times \omega_B) = T\tau_B \qquad (4.21)$$

$$I\dot{\omega}_B + \omega_B \times \mathrm{I}\omega_B = \tau_B \qquad (4.22)$$

Sviluppando i prodotti vettoriali che abbiamo mostrato prima, avremo quanto segue:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ry - pw \\ \dot{w} + pv - qu \end{bmatrix} = F_B \qquad (4.23)$$

In termini angolari si ha:

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = m \begin{bmatrix} \dot{p}I_x + qr(I_z - I_y) \\ \dot{q}I_y + pr(I_x - I_z) \\ \dot{r}I_z + pq(I_y - I_x) \end{bmatrix} = \tau_B \quad (4.24)$$

F_B è il vettore che rappresenta le forze esterne, mentre τ_B è il vettore che rappresenta i momenti esterni. Si osservi che, quanto finora mostrato, vale per un qualsiasi corpo rigido soggetto a delle forze ed a delle coppie.

Le forze esterne F_B possono essere scomposte in due termini:

1) un termine che rappresenta la forza gravitazionale F_B^g

2) un termine che rappresenta la forza di spinta (terminologia aeronautica "Thrust") F_B^T ,

$$F_B = F_B^g + F_B^T \qquad (4.25)$$

dove:

$$F_B^g = R^{-1} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \begin{bmatrix} -\sin(\theta) \\ \sin(\phi)\cos(\theta) \\ \cos(\phi)\cos(\theta) \end{bmatrix} mg \quad (4.26)$$

$$F_B^T = -\begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \quad (4.27)$$

La forza di spinta T è uguale a $T_i = b\Omega_i^2$, dove b è il coefficiente di spinta che dipende dal tipo di elica in questione, mentre Ω è la velocità angolare con cui gira l'elica. Il valore del coefficiente b si calcola tramite la seguente formula:

$$b = \frac{F_t}{\omega^2} \quad (4.28)$$

Dove F_t è la forza di spinta applicata da un'elica espressa in N, Mentre ω^2è la velocità angolare espressa in rad/sec, allora avremo:

$$b = \frac{64.22}{(2673.49)^2} = 8.988 \times 10^{-6} \quad (4.29)$$

Avendo quattro motori, possiamo scrivere quindi:

$$T = \sum_{i=1}^{4} T_i \qquad T = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \quad (4.30)$$

In termini angolari sappiamo che, per imprimere una coppia, è sufficiente avere una differente velocità tra due motori sullo stesso asse. Questo avviene grazie all'inerzia dei motori e, in particolare, le coppie sono calcolabili come:

$$M_x = lb(\Omega_2^2 - \Omega_4^2) \quad (4.31)$$

$$M_y = lb(\Omega_1^2 - \Omega_3^2) \quad (4.32)$$

$$M_z = d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \quad (4.33)$$

Dove l è la distanza tra ogni motore ed il centro di massa, mentre d è il coefficiente di drag, che dipende dall'elica e dalla sua velocità di rotazione. Questo coefficiente si calcola tramite la seguente formula:

$$C_d = \frac{d}{\frac{1}{2}\rho V^2 S} \tag{4.34}$$

Dove C_d rappresenta la resistenza aerodinamica, mentre ρ è la densità del fluido (nel nostro caso, l'aria); V è la velocità del corpo rispetto al fluido indisturbato, infine S rappresenta l'area di riferimento (in questo caso, l'area che l'elica ricopre ruotando). C_d è un valore che si ottiene sperimentalmente, di conseguenza, il coefficiente di drag d, va ottenuto tramite tentativi durante la taratura dei PID.

Ora che si è compreso il rapporto tra le velocità di rotazione dei motori e i momenti che esse generano, si possono, sulla base di ciò, stabilire le quattro uscite che i controllori dovranno generare per porre il drone nella posizione e nell'assetto desiderato:

$$\begin{aligned} U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 &= lb(\Omega_2^2 - \Omega_4^2) \\ U_3 &= lb(\Omega_1^2 - \Omega_3^2) \\ U_4 &= d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{aligned} \tag{4.35}$$

Effettuato il cambio di variabili, si può definire la seguente matrice:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ lb & 0 & -lb & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{4.36}$$

Ora volendo costruire un controllore, con azione di controllo sul sistema, in termini di U_1, U_2, U_3, U_4 ed invertendo la matrice sopra riportata, è possibile risalire alla velocità dei motori. Si ha, appunto:

$$\begin{bmatrix}\Omega_1^2\\ \Omega_2^2\\ \Omega_3^2\\ \Omega_4^2\end{bmatrix} = \begin{bmatrix}\frac{1}{4b} & 0 & \frac{1}{2lb} & -\frac{1}{4d}\\ \frac{1}{4b} & -\frac{1}{2lb} & 0 & \frac{1}{4b}\\ \frac{1}{4b} & 0 & -\frac{1}{2lb} & -\frac{1}{4d}\\ \frac{1}{4b} & \frac{1}{2lb} & 0 & \frac{1}{4d}\end{bmatrix}\begin{bmatrix}U_1\\ U_2\\ U_3\\ U_4\end{bmatrix} \qquad (4.37)$$

4.2 CONTROLLO QUADRIROTORE

Si è utilizzato il controllo proporzionale-integrale-derivativo, in breve "controllo PID". Esso è un sistema di retroazione negativa ampiamente impiegato nei sistemi di controllo automatici, il più comune nell'industria, in particolare nella versione PI (senza azione derivativa)[1].

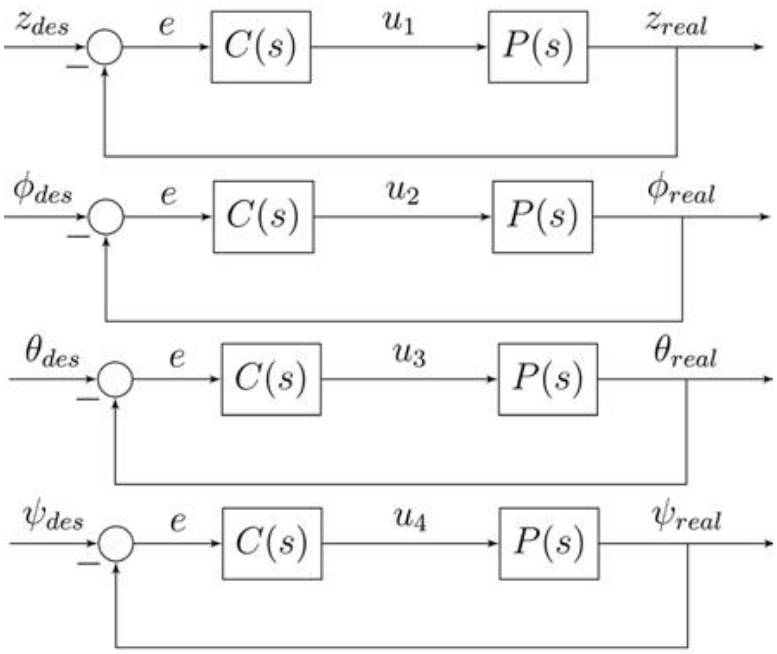
Grazie ad un input, che determina il valore attuale del sistema, è in grado di reagire ad un eventuale errore positivo o negativo, tendendo verso il valore zero. La reazione all'errore può essere regolata, ciò rende questo sistema molto versatile.

La legge di controllo è il seguente:

$$u(t) = K_P e(t) + K_D \frac{de(t)}{dt} + K_I \int e(t)dt \qquad (4.38)$$

Dove K_P, K_D e K_I sono i guadagni di ogni controllore e vengono tarati tramite alcuni metodi trattati nel capitol sucessivo.

In questo caso, cioè il quadrirotore, avranno quattro controllori PID, uno per ogni asse (Yaw, Pitch e Roll), più il controllo dell'altezza lungo l'asse Z.



$$u_1 = K_{p,z}(z_{des} - z_{real}) + K_{D,z}\frac{d(z_{des} - z_{real})}{dt} + K_I \int (z_{des} - z_{real})dt$$

$$u_2 = K_{p,z}(\phi_{des} - \phi_{real}) + K_{D,z}\frac{d(\phi_{des} - \phi_{real})}{dt} + K_I \int (\phi_{des} - \phi_{real})dt$$

$$u_3 = K_{p,z}(\theta_{des} - \theta_{real}) + K_{D,z}\frac{d(\theta_{des} - \theta_{real})}{dt} + K_I \int (\theta_{des} - \theta_{real})dt$$

$$u_4 = K_{p,z}(\psi_{des} - \psi_{real}) + K_{D,z}\frac{d(\psi_{des} - \psi_{real})}{dt} + K_I \int (\psi_{des} - \psi_{real})dt$$

Azione Proporzionale

Il termine “proporzionale” tiene conto del valore attuale dell'errore. Un valore di guadagno proporzionale alto fa sì che l'azione di controllo sia ampia, anche nel caso di errori meno rilevanti; mentre, un valore di guadagno proporzionale basso, rende meno considerevole il valore attuale dell'errore, privilegiando invece il valore che l'errore ha avuto in passato (azione integrale) e le dinamiche di variazione dell'errore nel tempo futuro (azione derivativa). In generale, per un valore di K_P troppo alto, il quadrirotore diventa altamente sensibile e cerca di correggere eccessivamente gli errori. In questi casi, la correzione dell'errore va troppo oltre, ed il quadrirotore oscilla ad alta frequenza. Per ridurre le oscillazioni

c'è la possibilità di diminuire il termine K_P ma non eccessivamente, altrimenti comincerebbe a risultare troppo lento nelle risposte.

Azione Integrale

Il termine "integrale" tiene appunto conto del valore assunto dall'errore in passato e fornisce l'azione necessaria per rimuovere l'errore di stato stazionario. Integra l'errore per un certo periodo di tempo fino a quando il valore dell'errore non raggiunge lo zero. Se diventasse troppo elevato, il quadrirotore apparirebbe molto rigido e non agile; inoltre, un valore eccessivo di K_i, potrebbe causare anche delle oscillazioni a bassa frequenza.

Azione Derivativa

Il termine "derivativo" serve a smorzare, cioè ridurre, l'eccessiva correzione causata dal termine proporzionale. Un elevato valore di K_d, può riportare delle vibrazioni nel quadrirotore, questo perché amplificherebbe il rumore all'interno del sistema e questo porterebbe al surriscaldamento dei motori. Un altro effetto collaterale di un K_d troppo alto è la diminuzione nella risposta del quadrirotore.

Di seguito si illustra uno **schema di controllo complessivo** del quadrirotore:

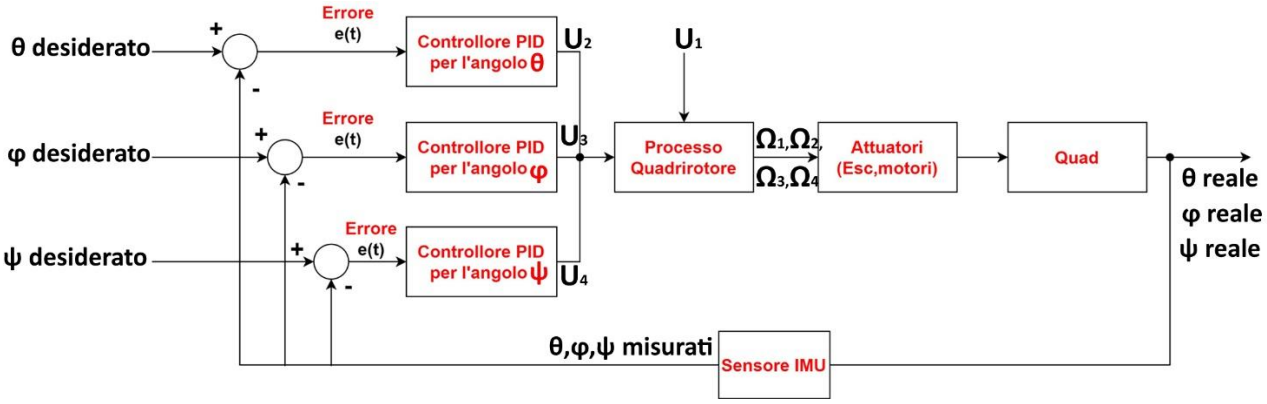


Fig 4.4: Schema complessivo di controllo quadrirotore

Nello schema 4.4 si noti che per il controllo dell'assetto, ci sono tre controllori PID. Invece, per la quota del quadrirotore non è stato impiegato un controllore PID in quanto a bordo del quadrirotore non esiste un sensore per il feedback. Quindi il controllo della quota avviene tramite l'ingresso U_1 che viene acquisito

dal telecomando tramite un segnale PWM poi viene trasformato in un dominio accettato dal modello matematico (4.37).

Negli sviluppi futuri si può installare un sensore di distanza a bordo del quadrirotore per avere un controllo della quota tramite un controllore PID

5 IMPLEMENTAZIONE SOFTWARE

In questo capitolo vengono illustrate le fasi di implementazione e di configurazione periferiche riguardanti la scheda “STM32H745ZI_Q”, utilizzando la piattaforma “STM32CubeIDE”, messa a disposizione da parte dell’azienda produttrice[18].

5.1 PIATTAFORMA “STM32CUBEIDE” E CREAZIONE NUOVO PROGETTO

STM32CubeIDE è una piattaforma di sviluppo C/C++ avanzata con configurazione periferica, generazione di codice, compilazione di codice e funzionalità di debug per microcontrollori e microprocessori STM32 .

Per la configurazione della scheda e la creazione di un nuovo progetto si esegue la seguente procedura:

1) dalla menu “File”--> “New”-->”STM32 Project”;
2) Si seleziona la scheda in questione (STM32H745ZI_Q);

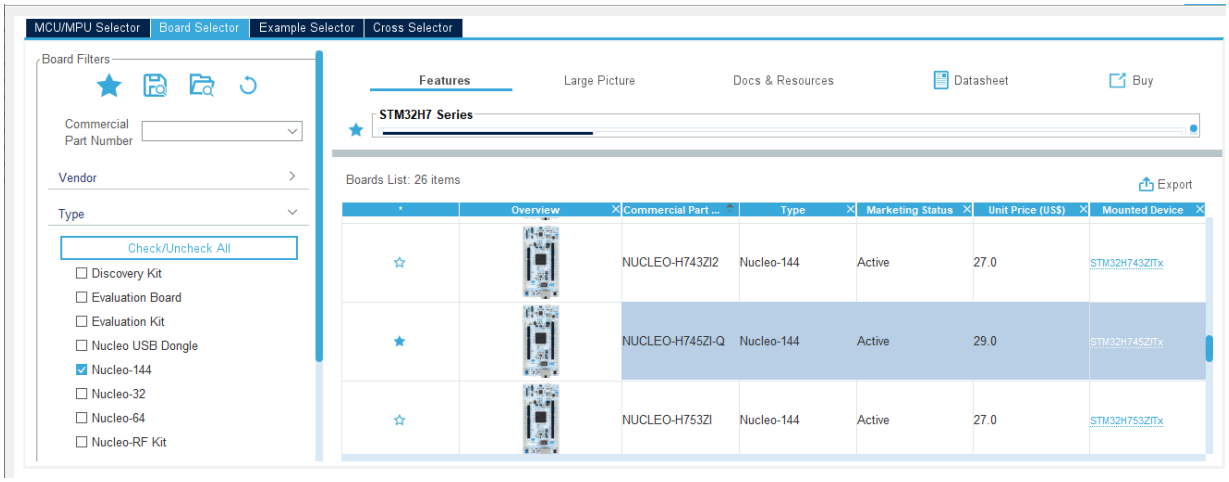


Fig.5.1: Configurazione scheda STM32

3) Si inserisce un nome per il progetto e si seleziona il linguaggio di programmazione. Si è deciso di utilizzare il linguaggio C, in quanto è più vicino al linguaggio macchina, quindi, torna utile durante la fase di implementazione.

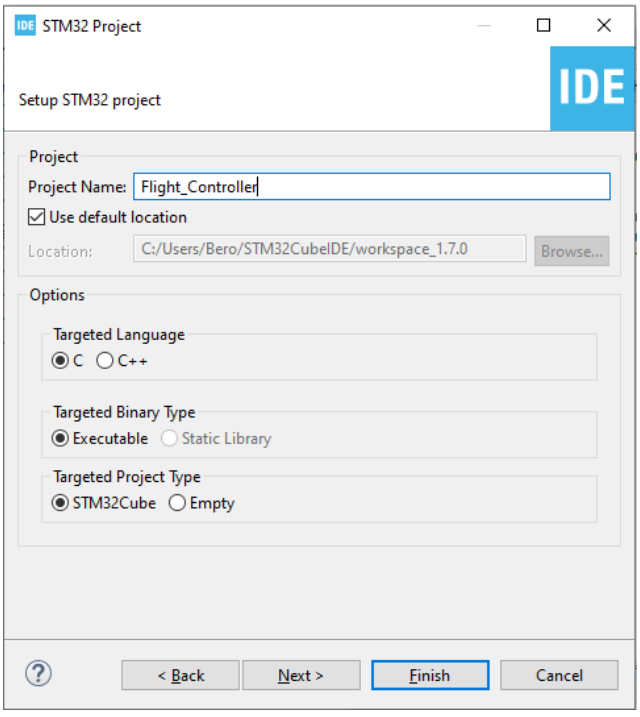


Fig.5.2: Impostazione nuovo progetto

Una volta eseguita questa procedura, viene creato il progetto e generato il codice di inizializzazione. In qualsiasi momento, durante lo sviluppo, si può tornare

all'inizializzazione e alla configurazione delle periferiche e rigenerare il codice di inizializzazione, senza alcun impatto sul codice già scritto.

5.2 DIAGRAMMA DI ATTIVITÀ E STRUTTURA COMPLESSIVA DEL SOFTWARE

Prima di cominciare lo sviluppo e l'implementazione del software, si è creato un diagramma di attività dove vengono illustrate le attività da eseguire da parte del microcontrollore per il controllo dell'assetto.

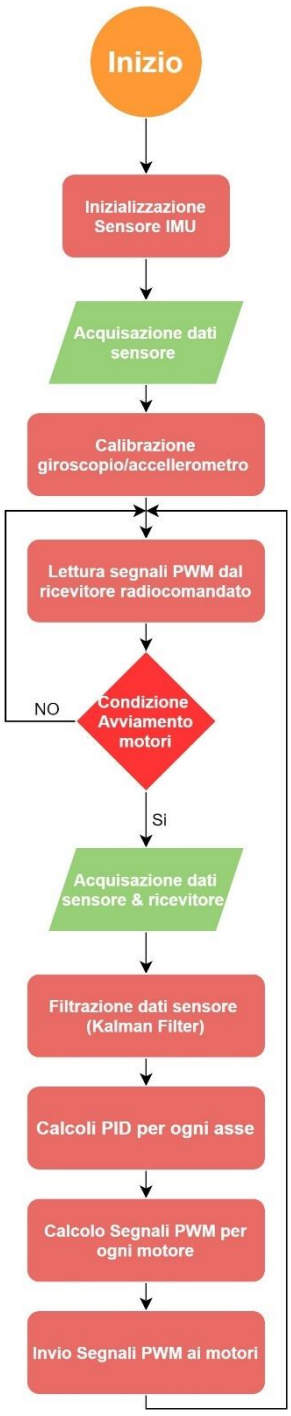


Fig.5.3: diagramma di attività

Di seguito viene illustrata la struttura complessiva del software che, è divisa in due parti, la prima parte nella funzione **while** e la seconda parte nella funzione **HAL_TIM_PeriodElapsedCallback:**

```
while (1)
{
    //When yaw stick is in left position (step 1).
    if ( rcready==4 && ch_3 < 991 && ch_4 < 1059) flag = 1;
      //When yaw stick is back in the center position start the motors (step 2).
    if (flag == 1 && ch_3 < 991 && ch_4 > 1562) flag = 2;

    if (arm==0 && flag==1){

        //MotorsArm(&htim3, TIM3);//arm motors
        HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_1);
         TIM3->CCR1=9.5;
        HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_2);
        TIM3->CCR2=9.5;
        HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);
        TIM3->CCR3=9.5;
        HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_4);
        TIM3->CCR4=9.5;
             arm=1;
    }
    if (arm==1 && flag==2){

       HAL_TIM_Base_Start_IT(&htim6);//inizialize timer6 in interrupt mode
        arm=0;

    }
```

Fig 5.3.1 prima parte del codice nella funzione while

Nella figura 5.3.1 si noti che è stata implementata la fase di armamento dei motori e del loro avviamento.

L'armamento dei motori avviene spostando la levetta della spinta tutta a sinistra.

```
if (flag == 2 && ch_3 < 991 && ch_4 > 2056) {
    //HAL_TIM_Base_Stop_IT(&htim7); //inizialize timer7 in interrupt mode
    HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_1);
    HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_2);
    HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);
    HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_4);

}
```

Fig 5.3.2 seconda parte del codice nella funzione while

Nella figura 5.3.2 avviene il disarmamento dei motori spostando la levetta della spinta del telecomando tutta a destra.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){

    //verify if the intrrupt comes from timer7
    if(htim==&htim7){
        //Read data from IMU (gyro,accel)
        MPU6050_Read_All(&hi2c2, &MPU6050);
        degrees_roll=MPU6050.KalmanAngleY;
        degrees_pitch=MPU6050.KalmanAngleX;
        degrees_yaw=MPU6050.Gz;
        }
//verify if the intrrupt comes from timer6
    if(htim==&htim6){
    if (flag==2){

     float virtualInputs [4]; // Temporary storage for PID results

     double dt = (double) (HAL_GetTick() - timerPid) / 1000;
            timerPid = HAL_GetTick();

        virtualInputs[0] =  0;
        virtualInputs[1] = PID_Compute(degrees_pitch, desiredpitch, &pitchPid,dt);//pid pitch
        virtualInputs[2] = PID_Compute(degrees_roll,  desiredroll, &rollPid,dt);//pid roll
        virtualInputs[3] = PID_Compute(degrees_yaw, desiredyaw,  &yawPid,dt);//pid yaw
```

Fig 5.3.3 prima parte del codice nella funzione **HAL_TIM_PeriodElapsedCallback()**

Nella figura 5.3.3 avvengono due interrupt, uno dal timer7 che funziona a frequenza di 250Hz, e un altro dal timer6 che funziona a una frequenza di 100 Hz.

All'avvenuta dell'interrupt del timer7, vengono acquisiti e filtrati i dati dal sensore inerziale tramite la funzione **MPU6050_Read_All(&hi2c2, &MPU6050).** In seguito vengono assegnati questi dati alle variabili **degrees_roll** e **degrees_pitch.** L'angolo yaw non viene filtrato quindi viene assegnato direttamente alla variabile **degrees_yaw.**

Invece, all'avvenuta dell'interrupt da parte del timer6 viene calcolato PID tramite la funzione **PID_Compute().**

```
    float* Speeds;

    //computes motor speeds (B_4 is for 4-cell battery, if you use a 3-cell, change it with B_3)

    Speeds = SpeedCompute (virtualInputs, B_3, L, D);


    PWM_1 = map(*(Speeds+0), 0, MOTOR_MAX_SPEED_3, MOTOR_MIN_UP, throttle);
    PWM_2 = map(*(Speeds+1), 0, MOTOR_MAX_SPEED_3, MOTOR_MIN_UP,throttle);
    PWM_3 = map(*(Speeds+2), 0, MOTOR_MAX_SPEED_3, MOTOR_MIN_UP, throttle);
    PWM_4 = map(*(Speeds+3), 0, MOTOR_MAX_SPEED_3, MOTOR_MIN_UP, throttle);


    TIM3->CCR1=PWM_1; //send pwm signal to motor 1
    TIM3->CCR2=PWM_2;//send pwm signal to motor 2
    TIM3->CCR3=PWM_3;//send pwm signal to motor 3
    TIM3->CCR4=PWM_4;//send pwm signal to motor 4
}
    //to stop motors during the test
if(flag==3) {
        TIM3->CCR1=0;
        TIM3->CCR2=0;
        TIM3->CCR3=0;
        TIM3->CCR4=0;
    }

}
```

Fig 5.3.4 seconda parte del codice nella funzione **HAL_TIM_PeriodElapsedCallback()**

Nella figura 5.3.4 vengono calcolati le velocità dei motori tramite la funzione **SpeedCompute()**che, prende come parametri l'uscita dei controllori PID visiti prima nella figura 5.3.3. Questa funzione contiene il modello matematico del quadrirotore e che sarà spiegata meglio nei prossimi capitoli.

Una volta calcolate le velocità dei motori vengono poi trasformati in un range tra 11 e **throttle** (spinta acquisita dal telecomando vedi figura 5.26) per essere poi inviati ai motori con dei segnali PWM.

5.3 CONFIGURAZIONE SENSORE "MPU6050" E ACQUISIZIONE DATI

La trasmissione dei dati tra il microcontrollore e il sensore avviene tramite il protocollo di comunicazione I2C.

Caratteristiche comunicazione I2C

I2C è un protocollo di comunicazione seriale sincrono che permette a molti dispositivi di scambiarsi dati fra loro utilizzando solo due cavi, uno per i dati ed uno per sincronizzare la comunicazione [12].

Il bus I2C è composto in totale da 4 cavi:

Linea	Descrizione
Vdd	Tensione di riferimento
SCL	Clock per la sincronizzazione della comunicazione
SDA	Linea su cui transitano i dati
GND	Massa di riferimento

Tabella 5.1 Descrizione protocollo I2C

I limiti superiori della velocità di clock del bus sono di 100 KHz, ma esistono anche varianti a 400 KHz e 3.4 MHz. Non esiste un limite inferiore, quindi, è possibile impostare un clock di 1 Hz. A questo bus possono essere collegati più dispositivi, che potranno comunicare tra loro assumendo di volta in volta uno il ruolo di master ed uno il ruolo di slave.

Nella comunicazione seriale, il master controllerà sempre il segnale SCL e scambierà i dati con lo slave tramite il segnale SDA. Una comunicazione avviene all'interno di due sequenze di segnali che il master invierà sul bus I^2C, queste sono il comando di START e di STOP. La regola fondamentale è che nella trasmissione dei dati, il segnale SDA può cambiare stato soltanto mentre il segnale SCL è basso, ed il suo valore viene poi letto durante il fronte di salita o di discesa del segnale SCL che diviene alto. Se il segnale SDA cambia valore mentre SCL è alto, significa che il master non sta trasmettendo o ricevendo un dato, ma sta iniziando o terminando la comunicazione, attraverso appunto i comandi di start e di stop.

Svantaggi comunicazione I2C

La comunicazione I2C ha alcuni limiti, tra i quali: la bassa velocità di comunicazione, richiede resistenze di pull-up, il che si traduce in un maggiore consumo di energia. Per sviluppi futuri si può utilizzare la comunicazione seriale SPI (Serial Peripheral Interface) che descrive una comunicazione singolo Master e singolo Slave, è di tipo sincrono e full-duplex ed è possibile sia trasmettere che ricevere dati in contemporanea.

La configurazione e l'attivazione della periferica I2C sulla scheda avviene tramite il file "NomeProgetto.ioc", seguendo la seguente procedura:

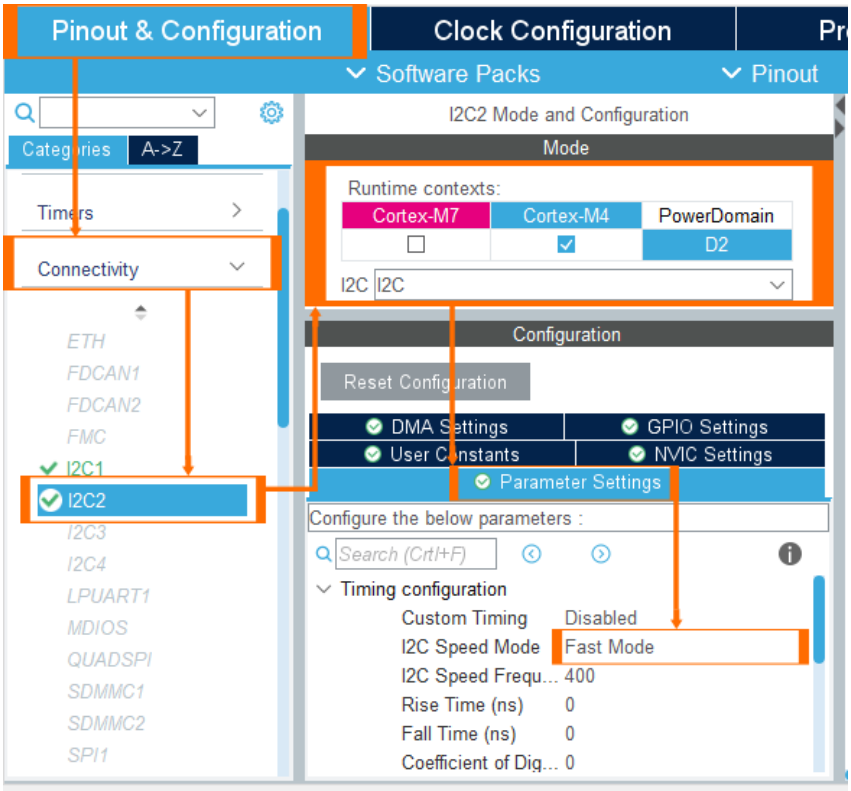


Fig.5.4: Attivazione e settaggio parametri periferica I2C

Una volta attivata la periferica, nella sezione “GPIO Settings” vengono specificati automaticamente i pin di ingresso per SDA e SCL.

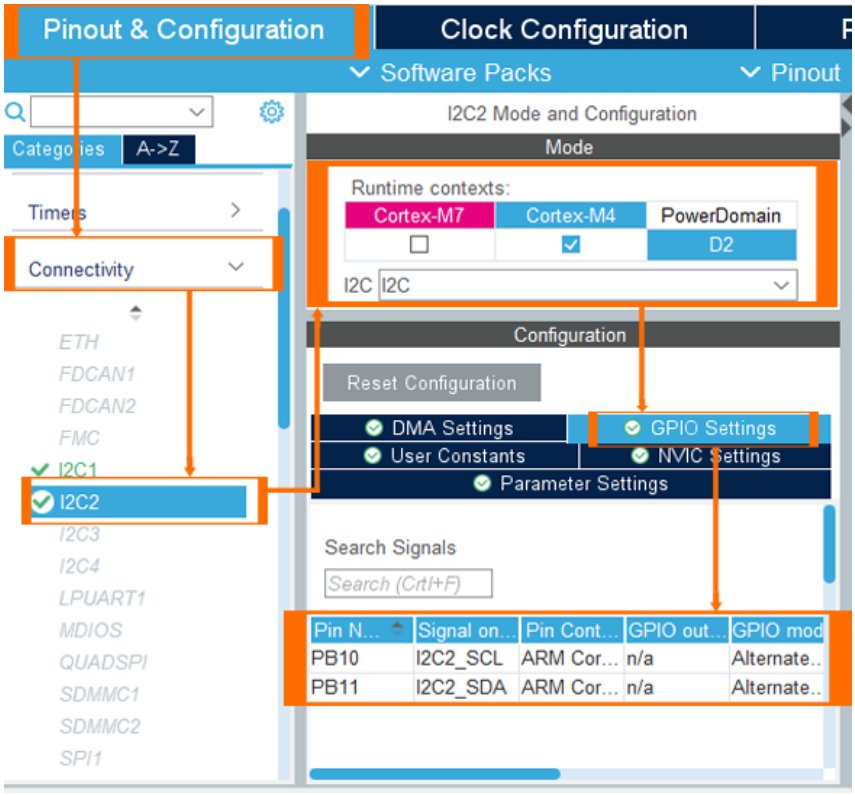


Fig.5.5: i Pin per il collegamento di SDA e SCL

Dopo il salvataggio di queste impostazioni viene creato, in modo automatico nel file “main.c”, il codice delle configurazioni appena fatte.

Per acquisire i dati dal sensore, si deve prima inizializzare con la seguente parte di codice:

```c
uint8_t MPU6050_Init(I2C_HandleTypeDef *I2Cx) {
    uint8_t check;
    uint8_t Data;

    // check device ID WHO_AM_I

    HAL_I2C_Mem_Read(I2Cx, MPU6050_ADDR, WHO_AM_I_REG, 1, &check, 1, i2c_timeout);

    if (check == 104)  // 0x68 will be returned by the sensor if everything goes well
    {
        // power management register 0X6B we should write all 0's to wake the sensor up
        Data = 0;
        HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, PWR_MGMT_1_REG, 1, &Data, 1, i2c_timeout);

        // Set DATA RATE of 1KHz by writing SMPLRT_DIV register
        Data = 0x07;
        HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, SMPLRT_DIV_REG, 1, &Data, 1, i2c_timeout);

        // Set accelerometer configuration in ACCEL_CONFIG Register
        // XA_ST=0,YA_ST=0,ZA_ST=0, FS_SEL=0 -> ± 2g
        Data = 0x00;
        HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, ACCEL_CONFIG_REG, 1, &Data, 1, i2c_timeout);

        // Set Gyroscopic configuration in GYRO_CONFIG Register
        // XG_ST=0,YG_ST=0,ZG_ST=0, FS_SEL=0 -> ± 250 °/s
        Data = 0x00;
        HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, GYRO_CONFIG_REG, 1, &Data, 1, i2c_timeout);
        return 0;
    }
    return 1;
}
```

Fig.5.6: Codice inizializzazione sensore “MPU6050”

Nella prima parte del codice si deve verificare che il sensore sia connesso correttamente, in questo modo si può avviare la procedura di inizializzazione scrivendo nei vari registri. È stato settato il fattore di sensibilità del giroscopio a ±250°/s, quindi, facendo riferimento al manuale Fig 5.7, va scritto il valore 0 all’indirizzo 0x1B.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
GYROSCOPE SENSITIVITY					
Full-Scale Range	FS_SEL=0		±250		°/s
	FS_SEL=1		±500		°/s
	FS_SEL=2		±1000		°/s
	FS_SEL=3		±2000		°/s

Fig 5.7 sensibilità giroscopio del “MPU6050”

Invece, per quanto concerne l’accelerometro, è stato settato il fattore di sensibilità a 2g, facendo sempre riferimento al manuale in Fig 5.8 e andando a scrivere il valore 0 all’indirizzo 0x1C.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
ACCELEROMETER SENSITIVITY					
Full-Scale Range	AFS_SEL=0		±2		*g*
	AFS_SEL=1		±4		*g*
	AFS_SEL=2		±8		*g*
	AFS_SEL=3		±16		*g*

Fig 5.8 sensibilità accelerometro del "MPU6050"

Inizializzato il sensore, si possono acquisire i dati relativi alla velocità angolare e l'accelerazione di ogni asse in gradi al secondo.

La velocità angolare di ciascun asse viene acquisita eseguendo il seguente codice:

```
void MPU6050_Read_Gyro(I2C_HandleTypeDef *I2Cx, MPU6050_t *DataStruct) {
    uint8_t Rec_Data[6];

    // Read 6 BYTES of data starting from GYRO_XOUT_H register

    HAL_I2C_Mem_Read(I2Cx, MPU6050_ADDR, GYRO_XOUT_H_REG, 1, Rec_Data, 6, i2c_timeout);

    DataStruct->Gyro_X_RAW = (int16_t) (Rec_Data[0] << 8 | Rec_Data[1]);
    DataStruct->Gyro_Y_RAW = (int16_t) (Rec_Data[2] << 8 | Rec_Data[3]);
    DataStruct->Gyro_Z_RAW = (int16_t) (Rec_Data[4] << 8 | Rec_Data[5]);

    /*** convert the RAW values into dps (°/s)
         we have to divide according to the Full scale value set in FS_SEL
         I have configured FS_SEL = 0. So I am dividing by 131.0
         for more details check GYRO_CONFIG Register              ****/

    DataStruct->Gx = DataStruct->Gyro_X_RAW / 131.0;
    DataStruct->Gy = DataStruct->Gyro_Y_RAW / 131.0;
    DataStruct->Gz = DataStruct->Gyro_Z_RAW / 131.0;
}
```

Fig 5.9: acquisizione velocità angolare

Nella parte di codice riportata sopra, vengono letti i valori RAW della velocità angolare dei tre assi dall'indirizzo 0x43, ogni asse è composto da 16 bit, quindi, in totale, 6 byte. Di conseguenza, questi valori RAW vengono divisi per 131, questo per ottenere, in definitiva, i valori effettivi della velocità angolare. Il numero su cui vengono divisi i valori RAW dipende dal coefficiente di sensibilità scelto in precedenza, figura 5.7 (FS_SEL=0).

Gyroscope ADC Word Length					
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)
	FS_SEL=1		65.5		LSB/(°/s)
	FS_SEL=2		32.8		LSB/(°/s)
	FS_SEL=3		16.4		LSB/(°/s)

Fig 5.10 divisore del valore RAW per il giroscopio

I valori RAW di accelerazione di ogni asse vengono letti dall'indirizzo 0x3B e vengono divisi per 16384.

ADC Word Length	Output in two's complement format				
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*
	AFS_SEL=1		8,192		LSB/*g*
	AFS_SEL=2		4,096		LSB/*g*
	AFS_SEL=3		2,048		LSB/*g*

Fig 5.11 divisore del valore RAW per l'accelerometro

L'acquisizione dei dati dal sensore avviene con una frequenza di 250 Hz, perciò, è stato attivato il Timer7 in modalità interrupt tramite i seguenti passaggi:

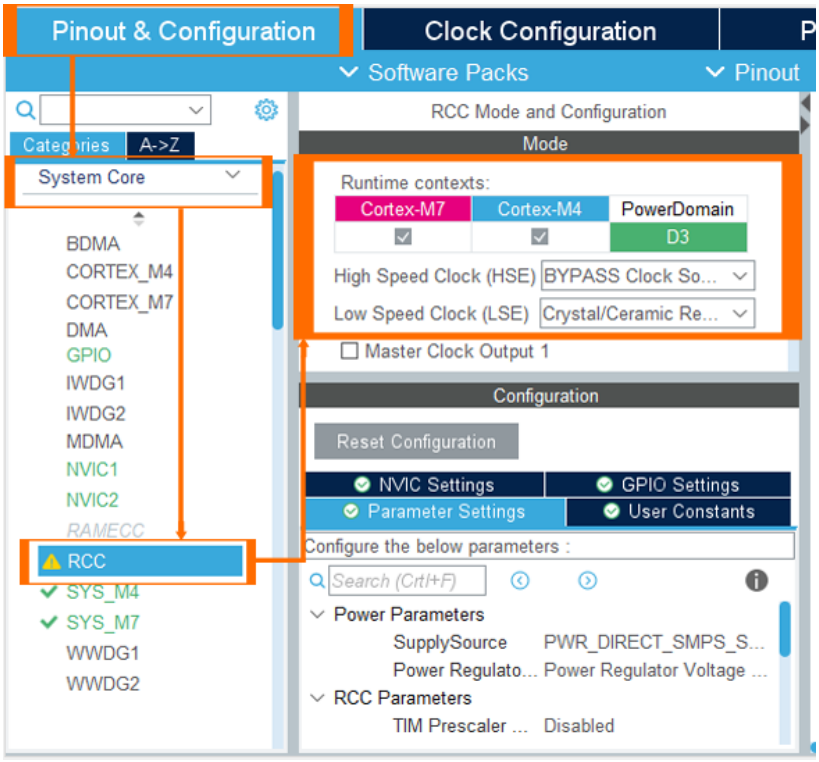


Fig 5.12 Configurazione RCC

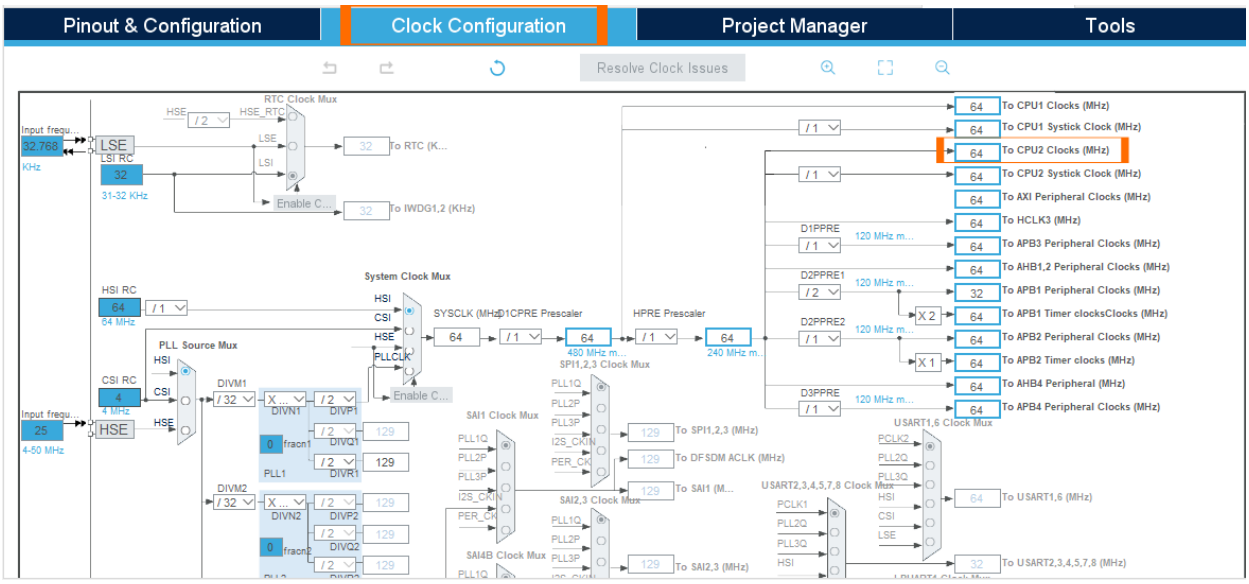


Fig 5.13 Configurazione Clock

Per settare la frequenza del timer si utilizza la seguente formula:

$$Update\ Event = \frac{Timer_{clock}}{(Prescaler+1)(Period+1)}$$

Quindi, per avere una frequenza di 250 Hz:

$$Update\ Event = \frac{64000000}{(6400+1)(40+1)} = 250\ Hz = 4\ ms$$

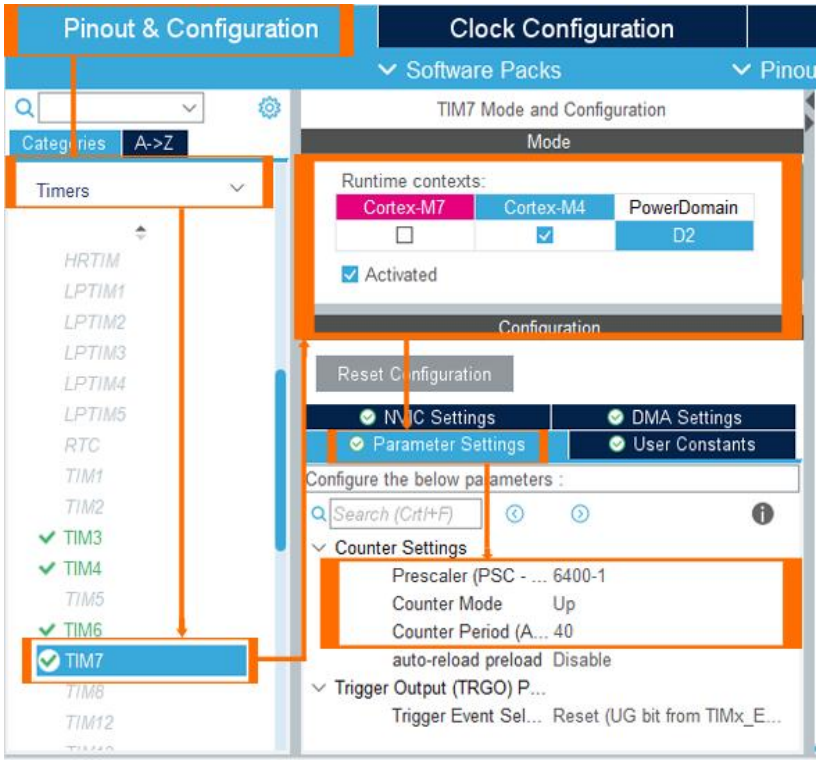


Fig 5.14: Attivazione e settaggio parametri Timer7

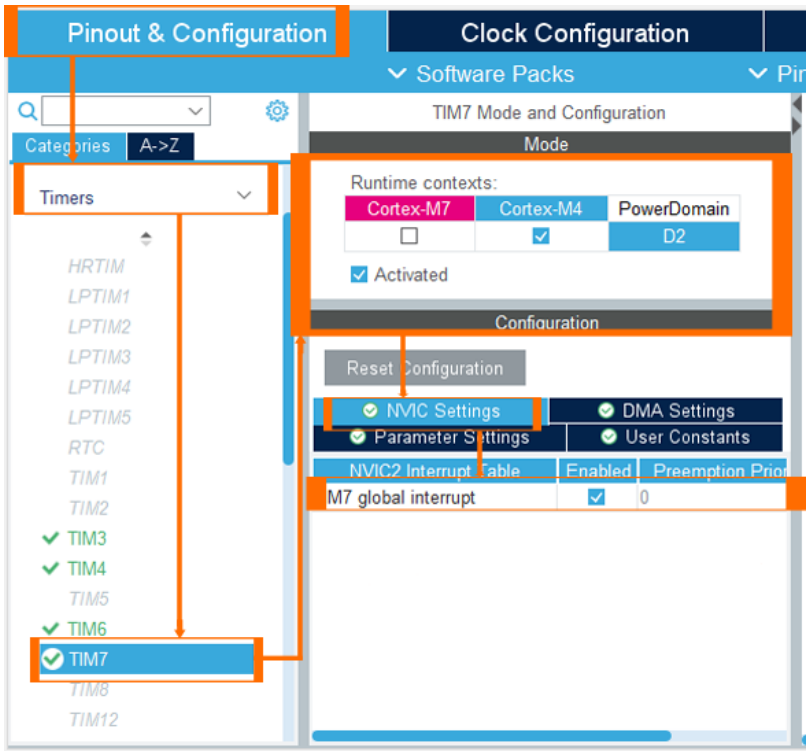


Fig 5.15: Attivazione modalità interrupt

Per attivare il timer nella modalità interrupt, esprimendolo in codice, si utilizza la funzione **HAL_TIM_Base_Start_IT(&htim7)** della libreria HAL. Invece intercettazione dell'avvenuto interrupt da parte del timer avviene tramite la funzione di callback **void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)**.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){

    //verify if the intrrupt comes from timer7
    if(htim==&htim7){
        //Read data from IMU (gyro,accel)
        MPU6050_Read_All(&hi2c2, &MPU6050);
```

Fig 5.16 codice acquisizione dati dal sensore ogni 4ms

5.4 CALIBRAZIONE GIROSCOPIO

Il giroscopio è un sensore che dev'essere calibrato per assicurarsene il corretto funzionamento. La sua calibrazione è necessaria affinché il valore della velocità angolare non derivi dal valore effettivo [9].

```
void calibirate_MPU6050(int num_iterations) {
  for (int i = 0; i < num_iterations; i++) {
      MPU6050_Read_Gyro(&hi2c2, &MPU6050);
    offset_gyx += MPU6050.Gyro_X_RAW;
    offset_gyy += MPU6050.Gyro_Y_RAW;
    offset_gyz += MPU6050.Gyro_Z_RAW;
    HAL_Delay(1);
  }

  offset_gyx /= num_iterations;
  offset_gyy /= num_iterations;
  offset_gyz /= num_iterations;
}
```

Fig 5.17: codice calibrazione giroscopio

La procedura di calibrazione consiste praticamente nel leggere dal sensore le velocità angolari un certo numero di volte, che varia tra 2000 e 3000, andando poi a calcolare la media dei valori acquisiti e dividendola per questo numero.

Il valore ottenuto dal calcolo precedente va poi sottratto, ogni volta, al valore della velocità angolare letto.

5.5 ACQUISIZIONE SEGNALI PWM DAL RICEVITORE RADIOCOMANDATO

Il ricevitore radiocomandato manda in uscita, su ogni canale, un segnale PWM di frequenza 50Hz. Il duty cycle del segnale PWM in uscita varia in base alla posizione della levetta del telecomando. Per controllare il quadrirotore servono solo quattro canali: spinta, Ptich, Roll e Yaw.

Fig 5.17: illustrazione funzionamento canali tra il telecomando e il ricevitore

Per l'acquisizione di questi quattro segnali PWM, sono stati configurati Timer4 e Timer5 in modalità "PWM Input Capture". Con questa modalità si attivano due canali di ingresso, uno per catturare il fronte di salita del segnale ed un altro per il fronte di discesa, come si vede nella Figura 5.18 [19].

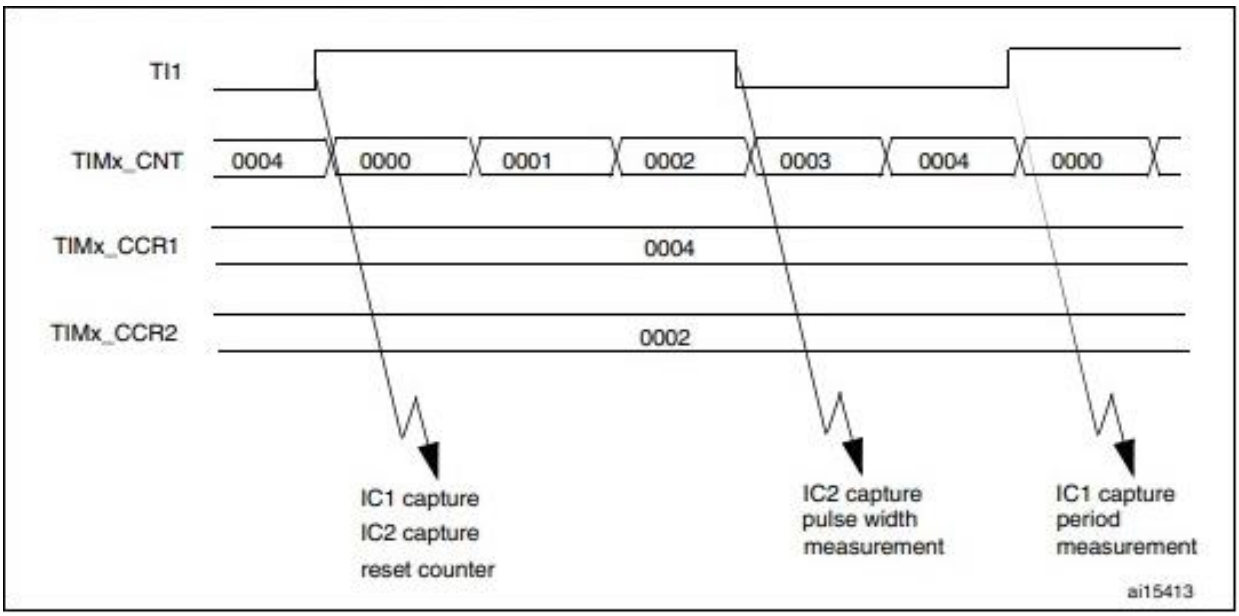


Fig 5.18: PWM Input Capture mode

Per l'attivazione della modalità "PWM input Capture" sulla scheda Stm32 si esegue la seguente procedura:

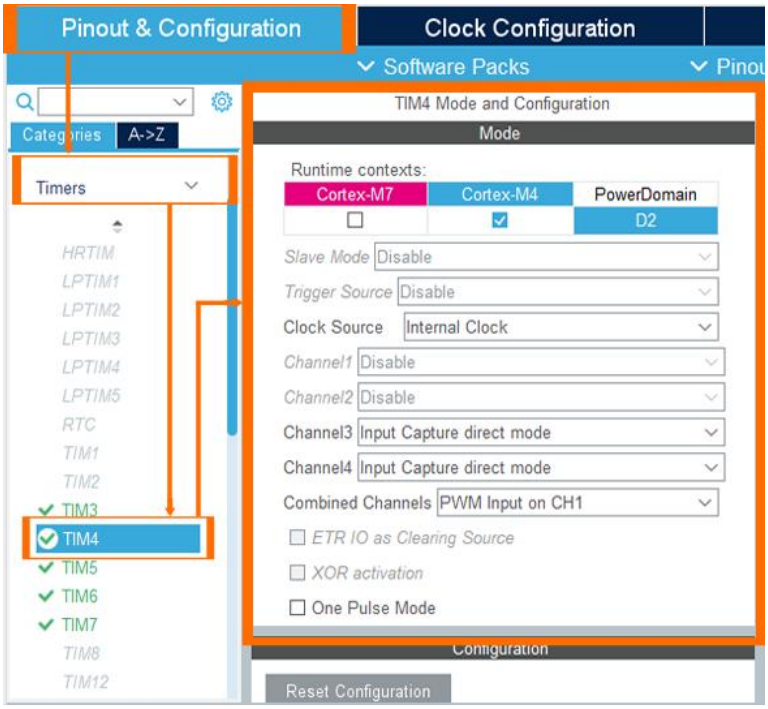


Fig 5.19 attivazione modalità "PWM input Capture" sul Timer4

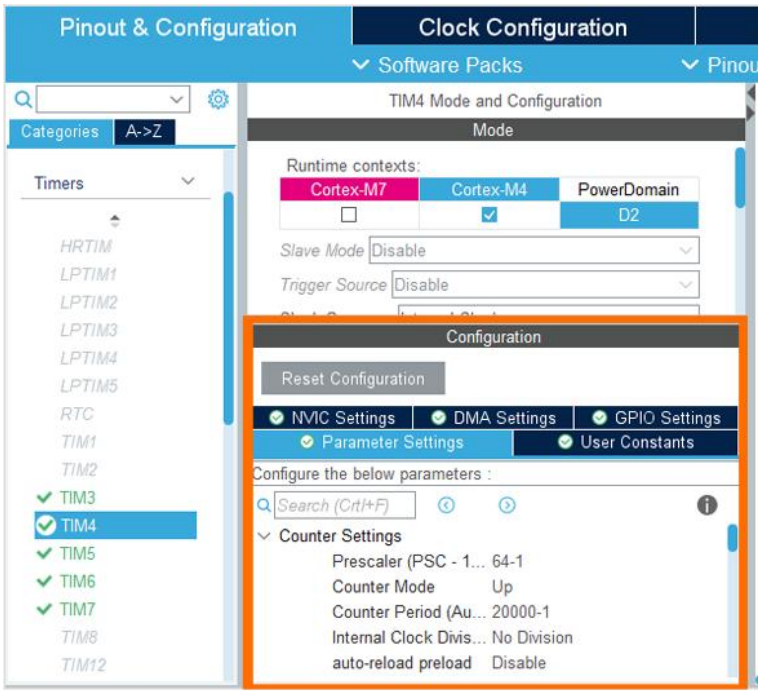


Fig 5.20 settaggio parametri Timer4

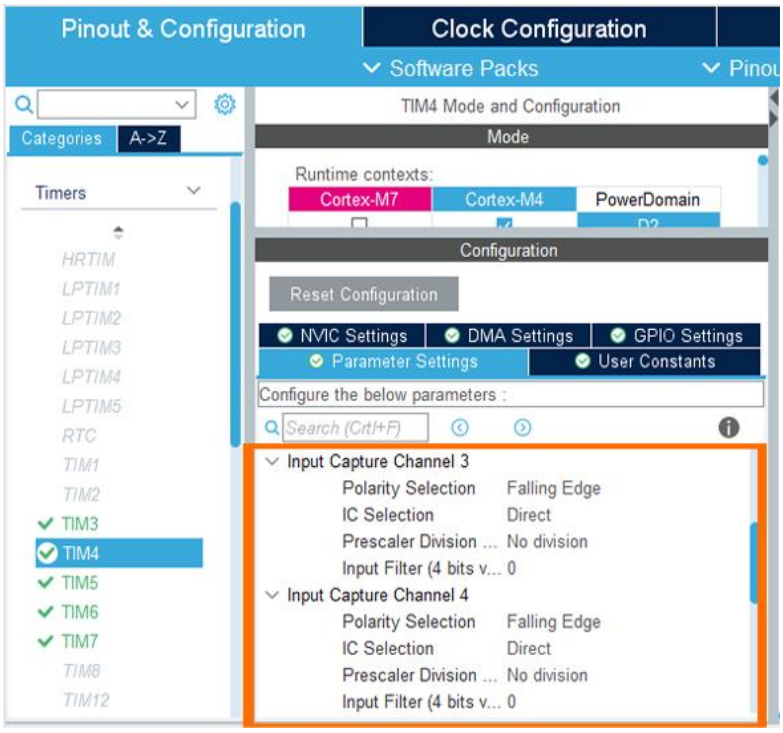


Fig 5.21 settaggio parametri dei canali numero 3 e 4

Si noti, nella figura 5.21, che i canali 3 e 4 sono stati settati per catturare il fronte di discesa del segnale, in quanto servirà in seguito per misurare la lunghezza di pulsazione (quando il segnale è alto).

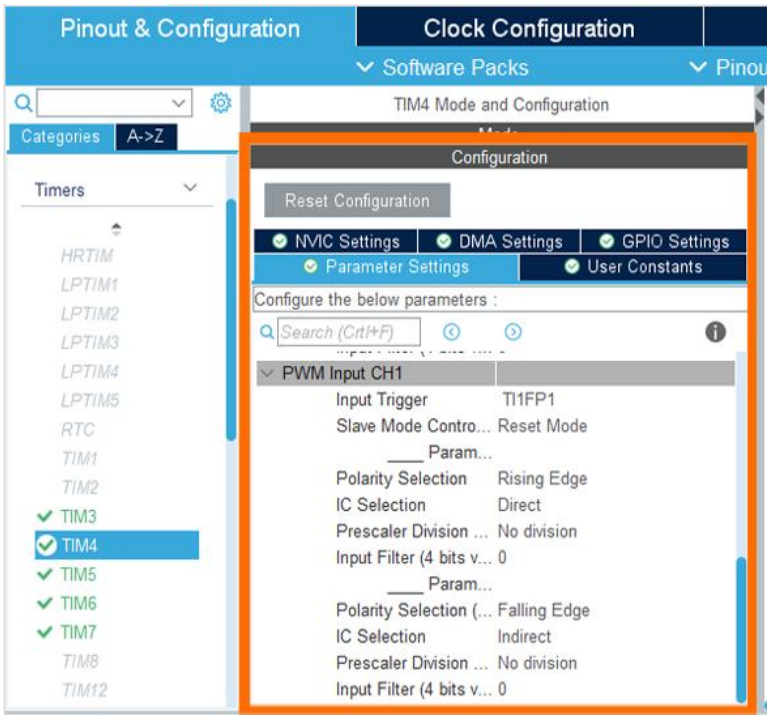


Fig 5.22 settaggio parametri dei canali numero 1 e 2

Invece, per i canali 1 e 2, nella figura 5.22, si noti che sono stati settati per catturare sia il fronte di salita che il fronte di discesa del segnale. La cattura del fronte di salita servirà per determinare il punto d'inizio e di fine del segnale.

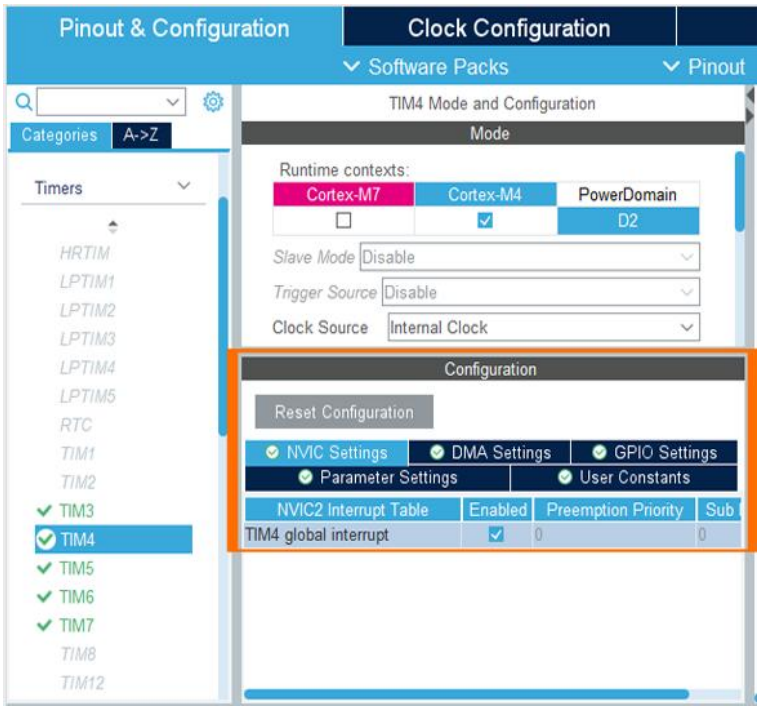


Fig 5.23 settaggio Timer4 nella modalità interrupt

Nella sezione “GPIO Settings” vengono specificati automaticamente i pin di ingresso dei canali.

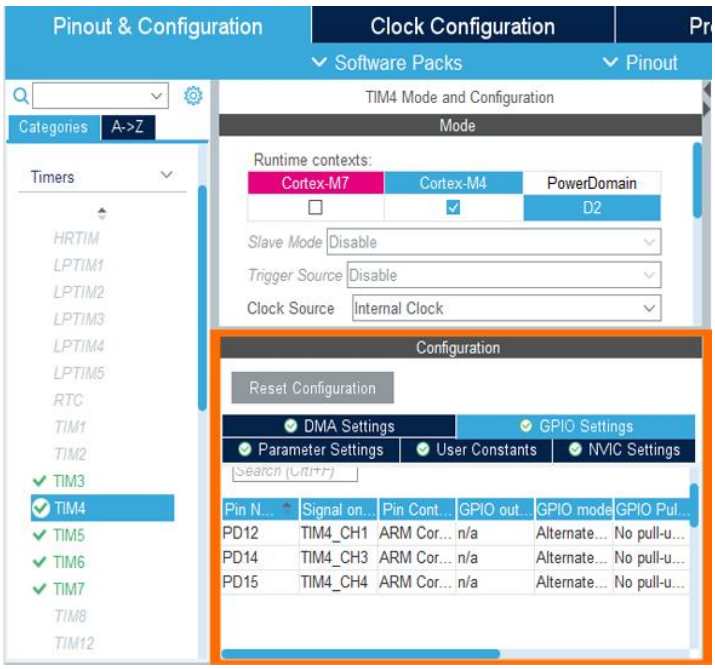


Fig 5.24: i Pin di ingresso segnale PWM

Ricapitolando, il **canale numero 1** serve per acquisire la lunghezza temporale del primo segnale PWM, mentre il **canale numero 2** acquisisce la sua lunghezza di pulsazione. Siccome tutti e quattro i segnali hanno la stessa lunghezza temporale di 20ms, allora gli altri **canali restanti (numero 3 e 4)** sono stati settati per leggere solamente la lunghezza di pulsazione del secondo e del terzo segnale.

Per quanto riguarda il quarto segnale sono stati attivati i canali 1 e 2 del timer5 seguendo la stessa procedura del timer4 appena illustrata.

Per attivare i timer nella modalità "PWM input Capture", esprimendolo in codice, si utilizza la funzione **HAL_TIM_IC_Start_IT(TIM_HandleTypeDef *htim, uint32_t Channel)** della libreria HAL.

```
HAL_TIM_IC_Start_IT(&htim4,TIM_CHANNEL_1);
HAL_TIM_IC_Start(&htim4,TIM_CHANNEL_2);
HAL_TIM_IC_Start(&htim4,TIM_CHANNEL_3);
HAL_TIM_IC_Start(&htim4,TIM_CHANNEL_4);

HAL_TIM_IC_Start_IT(&htim5,TIM_CHANNEL_1);
HAL_TIM_IC_Start(&htim5,TIM_CHANNEL_2);
```

Fig 5.25: Codice attivazione timers in modalità input Capture

Invece, l'intercettazione dell'avvenuto interrupt da parte dei timer avviene tramite la funzione di callback **void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim).**

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
    if( htim==&htim4 && htim->Channel==HAL_TIM_ACTIVE_CHANNEL_1){

        ch_1= HAL_TIM_ReadCapturedValue(htim,TIM_CHANNEL_2);//roll pin PD12
        desiredroll= map(ch_1, 1010, 2007, -15, 15);

        ch_2= HAL_TIM_ReadCapturedValue(htim,TIM_CHANNEL_3);//pitch pin PD14
        desiredpitch= map(ch_2, 988, 1987, -15, 15);

        ch_3= HAL_TIM_ReadCapturedValue(htim,TIM_CHANNEL_4);//throttle pin PD15
        throttle=map(ch_3, 987, 2000, 11, 20);
    }

    if(htim==&htim5 && htim->Channel==HAL_TIM_ACTIVE_CHANNEL_1){

        ch_4= HAL_TIM_ReadCapturedValue(&htim5,TIM_CHANNEL_2);//Yaw pin PA0
            desiredyaw= map(ch_4, 1057, 2057, -15, 15);
```

Fig 5.26 codice acquisizione segnali PWM

Nella figura 5.26 si noti che i segnali acquisiti dal ricevitore radiocomandato servono per controllare la spinta totale dei tutti e quattro i motori (l'altezza del quadrirotore) e gli angoli Pitch, Roll e Yaw desiderati.

Quindi il canale numero 1 del timer4 è stato settato per catturare il fronte di salita del primo segnale PWM (cioè la sua lunghezza temporale di 20 ms).Allora, nella parte di codice riportata sopra figura 5.26,quando avviene l'interrupt dal canale numero 1, significa che il segnale è stato inviato completamente (cioè sono passati 20 ms). A questo punto, siccome serve la lunghezza di pulsazione del segnale (quando il segnale è alto), si va a leggere dal canale numero 2, che è stato settato per catturare il fronte di discesa, la sua lunghezza di pulsazione. La stessa cosa succede con il secondo e il terzo segnale PWM, però siccome tutti i quattro segnali sono sincroni e hanno la stessa lunghezza temporale di 20 ms, non serve allora misurare la loro lunghezza temporale. Infatti, si va direttamente ad acquisire la loro lunghezza di pulsazione (quando il segnale è alto),dal canale numero 3 e 4 del timer4.Infine, visto che non ci sono più canali disponibili del timer4, sono stati settati i canali numero 1 e 2 del timer5 per l'acquisizione del quarto segnale.

La funzione **map()** serve per spostare il dominio, in questo caso per trasformare i segnali di PWM letti in angoli.

```
float map(float val, float from_src, float to_src, float from_dst, float to_dst)
{
    return (((to_dst-from_dst)/(to_src-from_src))*(val-from_src)) + from_dst;
}
```

Fig 5.26 codice della funzione map().

La funzione map() prende i seguenti parametri:

- **val**: il valore della variabile da cambiare il suo dominio;
- **from_src**: il valore minimo che può assumere la variabile **val;**
- **to_src**: il valore massimo che può assumere la variabile **val;**
- **from_dst**: il valore minimo che si vuole che la variabile **val** assumi (il valore minimo del nuovo dominio);
- **to_dst**: il valore massimo che si vuole che la variabile **val** assumi (il valore massimo del nuovo dominio).

Infine, la funzione tramite un calcolo matematico restituisce il valore della variabile **val** nel nuovo dominio.

Quindi il primo segnale, usando la funzione **map()**, viene trasformato nel angolo desiderato di Roll che può variare tra -15 e +15 gradi. La stessa funzione su utilizza per l'angolo Pitch e Roll. Invece, il terzo segnale viene trasformato in un valore che varia tra 11 e 20 che rappresenta la spinta minima e massima dei motori per il controllo dell'altezza.

Per non occupare tante periferiche della scheda Stm32, si può, invece che acquisire i segnali PWM di ogni canale, utilizzare l'uscita SBUS del ricevitore. SBUS è un protocollo di comunicazione che tramite l'utilizzo di solamente un cavo, è in grado di comunicare i valori di tutti i sei canali del ricevitore. Il lato negativo di questo protocollo è rappresentato da una bassa velocità nella comunicazione.

Il telecomando acquistato mette a disposizione fino a sei canali di controllo. Per il controllo del quadrirotore sono stati utilizzati quattro canali; quindi, gli altri due canali restanti si possono usare per manipolare altri movimenti o per cambiare tra le varie modalità di volo del quadrirotore.

Per sviluppi futuri si possono acquistare altri tipi di telecomandi che mettono a disposizione uno schermo per monitorare lo stato del quadrirotore, ad esempio: l'assetto, l'altezza dal suolo e la percentuale di batteria.

5.6 FILTRO KALMAN

Il filtro Kalman opera tramite un algoritmo che utilizza una serie di misurazioni osservate nel tempo, quindi, nel caso del quadrirotore, lette dall'accelerometro e dal giroscopio. Queste misurazioni contengono rumore, che dunque contribuisce all'errore delle stesse. Il filtro Kalman stima lo stato del sistema sulla base degli stati attuali e precedenti, che tendono ad essere più precisi rispetto alle sole misurazioni.

L'accelerometro è in generale molto rumoroso quando viene utilizzato per misurare l'accelerazione gravitazionale, poiché il quadrirotore è in continuo movimento. Mentre il valore del giroscopio devia da quello effettivo. In poche parole, il giroscopio risulta più adatto per un periodo di tempo breve, poiché risponde più velocemente alle variazioni. Invece, l'accelerometro è più adatto per

un lungo periodo di tempo, in quanto torna sempre al valore effettivo e non devia.

In realtà esiste un modo semplice per affrontare questo problema utilizzando il filtro complementare, ma questo filtro non è accurato come il filtro Kalman.

Il filtro di Kalman opera producendo una stima statisticamente ottimale dello stato del sistema in base alle misurazioni. Di seguito vengono illustrate le equazioni del filtro e le modalità di implementazione di esse [6][7].

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{x}_{k-1|k-1} + \boldsymbol{B}\dot{\theta}_k$$

$$\begin{bmatrix}\theta \\ \dot{\theta}_b\end{bmatrix}_{k|k-1} = \begin{bmatrix}1 & -\Delta t \\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}_b\end{bmatrix}_{k-1|k-1} + \begin{bmatrix}\Delta t \\ 0\end{bmatrix}\dot{\theta}_k$$

$$= \begin{bmatrix}\theta - \dot{\theta}_b\Delta t \\ \dot{\theta}_b\end{bmatrix}_{k-1|k-1} + \begin{bmatrix}\Delta t \\ 0\end{bmatrix}\dot{\theta}_k$$

$$= \begin{bmatrix}\theta - \dot{\theta}_b\Delta t + \dot{\theta}\Delta t \\ \dot{\theta}_b\end{bmatrix}$$

$$= \begin{bmatrix}\theta + \Delta t(\dot{\theta} - \dot{\theta}_b) \\ \dot{\theta}_b\end{bmatrix}$$

Lo stato del sistema al tempo k è dato da:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{x}_{k-1|k-1} + \boldsymbol{B}\dot{\theta}_k$$

Dove X_k è la matrice di stato che è data da:

$$\begin{bmatrix}\theta \\ \dot{\theta}_b\end{bmatrix}_{k|}$$

Come si può notare che l'uscita del filtro sarà l'angolo θ ma anche il bias $\dot{\theta}$ basato sulle misure dell'accelerometro e del giroscopio. Il bias è la quantità di spostamento del giroscopio. Ciò significa che si può ottenere la velocità reale sottraendo il bias dalla misurazione del giroscopio.

La matrice F è il modello di transizione di stato applicato allo stato precedente X_{k-1}.

In questo caso F è definito come:

$$\begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$$

La matrice B è il modello di input di controllo, che è definita come:

$$\begin{bmatrix} \Delta t \\ 0 \end{bmatrix}$$

Questa parte viene tradotta in linguaggio c nel seguente codice:

```
double rate = newRate - Kalman->bias;
Kalman->angle += dt * rate;
```

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^T + \boldsymbol{Q}_k$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} & P_{01} - \Delta t P_{11} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) + Q_\theta \Delta t & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix}$$

$$= \begin{bmatrix} P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix}$$

Nella funzione riportata sopra si prova a stimare la matrice di covarianza degli errori priori P [k | k-1] basata sulla precedente matrice di covarianza degli errori P [k-1 | k-1]. Questa matrice viene utilizzata per stimare quanto si fida dei valori correnti dello stato stimato. Più piccolo è più che si fida dello stato attuale stimato. Il principio dell'equazione sopra è in realtà abbastanza facile da capire, poiché è abbastanza ovvio che la covarianza dell'errore aumenterà dall'ultima volta che è stata aggiornata la stima dello stato; quindi, è stato moltiplicata la matrice della covarianza dell'errore per il modello di transizione di stato F e la trasposta di tale F^T e aggiungere il rumore di processo corrente Q_k all'istante k.

La matrice di covarianza degli errori P nel nostro caso è una matrice 2×2:

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}$$

L'equazione portata sopra viene scritta nel seguente modo:

```
Kalman->P[0][0] += dt * (dt * Kalman->P[1][1] - Kalman->P[0][1] - Kalman->P[1][0] + Kalman->Q_angle);
Kalman->P[0][1] -= dt * Kalman->P[1][1];
Kalman->P[1][0] -= dt * Kalman->P[1][1];
Kalman->P[1][1] += Kalman->Q_bias * dt;
```

$$\begin{aligned} \tilde{\boldsymbol{y}}_k &= \boldsymbol{z}_k - \boldsymbol{H}\hat{x}_{k|k-1} \\ &= \boldsymbol{z}_k - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} \\ &= \boldsymbol{z}_k - \theta_{k|k-1} \end{aligned}$$

Nell'equazione riportata sopra avviene la parte di aggiornamento dello stato.

All'inizio si calcola la differenza tra la misura Z_k e lo stato priori X_[k | k-1] (con stato priori si intende la stima della matrice di stato all'istante corrente k basata sullo stato precedente del sistema e le stime degli stati precedenti).

Il modello di osservazione H viene utilizzato per mappare lo stato a priori x_[k | k-1] nello spazio osservato che è la misura dell'accelerometro.

$$\begin{aligned} \boldsymbol{S}_k &= \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R} \\ &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \boldsymbol{R} \\ &= P_{00_{k|k-1}} + \boldsymbol{R} \\ &= P_{00_{k|k-1}} + var(v) \end{aligned}$$

In questo passaggio si calcola la covarianza dell'innovazione:

$$\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R}$$

Ciò che fa è cercare di prevedere quanto si deve fidare della misurazione basata sulla matrice di covarianza degli errori priori P_[k | k-1] e la matrice di covarianza di misura R. Il modello di osservazione H viene utilizzato per mappare la matrice di covarianza degli errori priori P_[k | k-1] nello spazio osservato.

Maggiore è il valore del rumore di misurazione, maggiore è il valore di S, ciò significa che non si fidia molto della misurazione.In questo caso S non è una matrice.

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{S}_k^{-1}$$

$$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \boldsymbol{S}_k^{-1}$$

$$= \begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1} \boldsymbol{S}_k^{-1}$$

$$= \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1}}{\boldsymbol{S}_k}$$

Il passo successivo è calcolare il guadagno di Kalman. Il guadagno di Kalman viene utilizzato per indicare quanto si fida dell'innovazione ed è definito come:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{S}_k^{-1}$$

Si può notare che se non si fida dell'innovazione, la covarianza dell'innovazione S avrà valore alto e se si fida della stima dello stato allora la matrice di covarianza degli errori P avrà valore basso. Il guadagno di Kalman avrà quindi un valore basso e opposto se si fidia l'innovazione ma non si fida della stima dello stato attuale.

Se lo stato iniziale non è noto è possibile impostare la matrice di covarianza degli errori in questo modo:

$$\boldsymbol{P} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

In questo caso il guadagno di Kalman è una matrice 2×1:

$$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}$$

Le equazioni portate sopra vengono scritte in C nel seguente modo:

```
double y = newAngle - Kalman->angle;

double S = Kalman->P[0][0] + Kalman->R_measure;

double K[2];
K[0] = Kalman->P[0][0] / S;
K[1] = Kalman->P[1][0] / S;
```

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \, \tilde{\boldsymbol{y}}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k} = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \tilde{\boldsymbol{y}}_k$$

$$= \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \, \tilde{\boldsymbol{y}} \\ K_1 \, \tilde{\boldsymbol{y}} \end{bmatrix}_k$$

Ora si può aggiornare la stima posteriori dello stato attuale (la stima dello stato al tempo k date le osservazioni fino al tempo k compreso):

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \, \tilde{\boldsymbol{y}}_k$$

Questo viene fatto aggiungendo lo stato a priori:

$$\hat{\boldsymbol{x}}_{k|k-1}$$

con il guadagno di Kalman moltiplicato per l'innovazione:

$$\tilde{\boldsymbol{y}}_k$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H})\boldsymbol{P}_{k|k-1}$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

$$= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 & 0 \\ K_1 & 0 \end{bmatrix}_k \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

$$= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{bmatrix}$$

Infine, l'ultimo passaggio è aggiornare la matrice di covarianza degli errori posteriori:

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H})\boldsymbol{P}_{k|k-1}$$

Dove I è la matrice di identità ed è definita come:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ricapitolando, il filtro fondamentalmente si auto_corregge la matrice di covarianza degli errori in base a quanto è stata corretta la stima. Questo ha senso poiché è stato corretto lo stato in base alla matrice di covarianza degli errori a priori P_[k | k-1], ma anche la covarianza dell'innovazione S_k.

In linguaggio C Le equazioni portate sopra vengono scritte in C nel seguente modo:

```
Kalman->angle += K[0] * y;
Kalman->bias += K[1] * y;

double P00_temp = Kalman->P[0][0];
double P01_temp = Kalman->P[0][1];

Kalman->P[0][0] -= K[0] * P00_temp;
Kalman->P[0][1] -= K[0] * P01_temp;
Kalman->P[1][0] -= K[1] * P00_temp;
Kalman->P[1][1] -= K[1] * P01_temp;

return Kalman->angle;
```

Di seguito viene illustrata la struttura dati **Struct_Kalman** che, serve per definire le matrici di covarianza R e Q, bias illustrato sopra e la matrice dello stato iniziale P.

```
typedef struct {
    double Q_angle;
    double Q_bias;
    double R_measure;
    double angle;
    double bias;
    double P[2][2];
} Kalman_t;
```

Singolarità degli angoli Pitch e Roll

Un problema principale del filtro Kalman per quanto riguarda gli angoli di Roll e Pitch è che il modello utilizza gli angoli di Eulero che contengono singolarità a ±90 gradi. Infatti, se il quadrirotore deve eseguire manovre acrobatiche, si consiglia di utilizzare quaternioni al posto degli angoli di Eulero per descrivere l'assetto. L'approccio del quaternione è più difficile da implementare, ma il modello del quaternione non ha singolarità a ±90◦.

5.7 CALCOLO PID

Per realizzare un controllore PID è sufficiente discretizzare la formula integro-differenziale che lo definisce.

Innanzitutto, è necessario calcolare l'errore attraverso la differenza tra l'uscita desiderata e quella attuale:

```
float error = setPoint - input;
```

Fig 5.27 codice calcolo errore

Successivamente si calcola il termine derivativo. Si approssima la derivata dell'errore con il metodo delle differenze finite:

($\frac{d(x)}{dt} \approx \frac{x_f - x_i}{T}$ dove T è il tempo di campionamento).

```
float dInput = (error - conf->lastError) / dt
```

Fig 5.28 codice calcolo termine derivativo

Il termine integrale si calcola approssimando l'errore con la formula composta della regola del rettangolo ($\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_i)$ con $h = \frac{b-a}{M}$, M è il numero di sottointervalli in cui è applicata la regola). Per il quadrirotore si pone h pari al periodo di campionamento dt e, non conoscendo in partenza l'ampiezza dell'intervallo, si somma progressivamente il prodotto e∗T.

```
conf->ITerm += (error * dt)* conf->ki;
```

Fig 5.29 codice calcolo termine integrale

A questo punto, non resta che moltiplicare ogni termine per il corrispettivo guadagno:

```
float output = (conf->kp * error) + (conf->ITerm) + (conf->kd * dInput);
```

Fig 5.30 codice calcolo PID

5.8 CALCOLO SEGNALI PWM PER OGNI MOTORE

Il calcolo dei segnali PWM che vanno poi inviati ai motori si ricavano applicando il modello matematico del quadrirotore trattato nel capitolo 4. Il modello matematico è stato tradotto in codice tramite la funzione **SpeedCompute** come segue:

```
float* SpeedCompute (float virtualInputs [], float b, float l, float d)
{

    static float Speeds_quad[4];
    static float Speeds[4];

        Speeds_quad[0] = (1/(4*b))*virtualInputs[0] - (1/(2*l*b))*virtualInputs[2] - (1/(4*d))*virtualInputs[3];
        Speeds_quad[1] = (1/(4*b))*virtualInputs[0] - (1/(2*l*b))*virtualInputs[1] + (1/(4*d))*virtualInputs[3];
        Speeds_quad[2] = (1/(4*b))*virtualInputs[0] + (1/(2*l*b))*virtualInputs[2] - (1/(4*d))*virtualInputs[3];
        Speeds_quad[3] = (1/(4*b))*virtualInputs[0] + (1/(2*l*b))*virtualInputs[1] + (1/(4*d))*virtualInputs[3];
```

Fig 5.31 modello matematico tradotto in codice

il vettore **virtualInput** rappresenta l'uscita del controllo PID

```
virtualInputs[1] = PID_Compute(degrees_pitch, desiredpitch, &pitchPid,dt);
virtualInputs[2] = PID_Compute(degrees_roll,  desiredroll, &rollPid,dt);
virtualInputs[3] = PID_Compute(degrees_yaw, desiredyaw,  &yawPid,dt);
```

Fig 5.32 uscita del calcolo PID

Mentre le variabili b,l e d si possono ottenere tramite il metodo spiegato nel capitolo 4.1

Poiché il modello matematico presuppone che la rotazione del motore possa essere invertita, non è possibile da attuare nella pratica, anche perché, nel passaggio da velocità al quadrato a velocità, si incorrerebbe nel calcolo di una radice di un numero negativo. Per ovviare a tutto ciò è stato ampliata la funzione nel modo seguente:

```
const float abs_speedQuad_MAX = (1/(4*b)) + 1/(2*l*b) + 1/(4*d); //maximum reachable squared speed

float speedQuad_min = - (1/(4*b)) -(1/(2*l*b)) - (1/(4*d));//lower bound for squared speed 0.
float speedQuad_Max =  (1/(4*b)) + 1/(2*l*b) + 1/(4*d);//upper bound for squared speed 0.
Speeds[0] = sqrt(map(Speeds_quad[0], speedQuad_min, speedQuad_Max, 0, abs_speedQuad_MAX));
Speeds[1] = sqrt(map(Speeds_quad[1], speedQuad_min, speedQuad_Max, 0, abs_speedQuad_MAX));
Speeds[2] = sqrt(map(Speeds_quad[2], speedQuad_min, speedQuad_Max, 0, abs_speedQuad_MAX));
Speeds[3] = sqrt(map(Speeds_quad[3], speedQuad_min, speedQuad_Max, 0, abs_speedQuad_MAX));

return Speeds;
```

Fig 5.33 calcolo della radice delle velocità

Ad ogni velocità al quadrato è stata applicata una trasformazione che sposta il suo dominio in uno esclusivamente positivo tramite la funzione **map()** (vedi figura 5.26).

Il nuovo intervallo [0, abs_speedQuad_MAX] è stato scelto considerando 0 come limite sinistro e il valore massimo, raggiungibile dal sistema di equazioni, derivante dalla matrice d'assetto come limite destro.

5.9 INVIO SEGNALI PWM AI MOTORI

I motori brushless si controllano tramite un segnale PWM di frequenza 50Hz (periodo 20ms). La velocità dei motori dipende dal duty cycle del segnale PWM inviato.

Per avviare questo tipo di motori, prima devono essere armati. L'armamento è una procedura di sicurezza che consiste nell'inviare un certo segnale PWM con un duty cycle del 4.75 %.

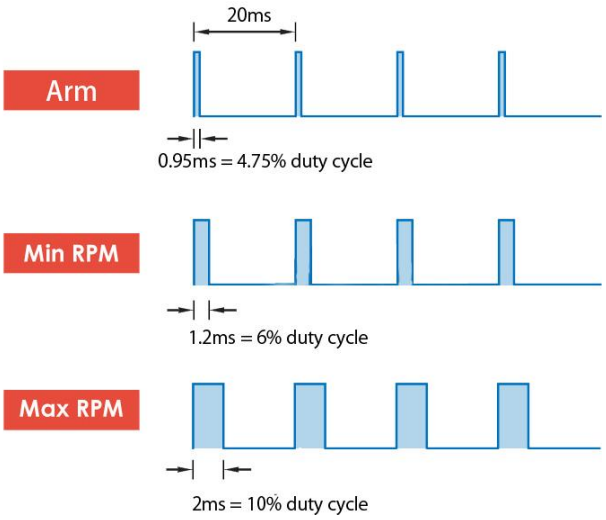


Fig 5.34: schema illustrativo per il controllo del motore

Per la generazione dei segnali PWM dalla scheda STM32, sono stati configurati i canali del Timer3 in modalità “PWM Generation” con la seguente procedura[18]:

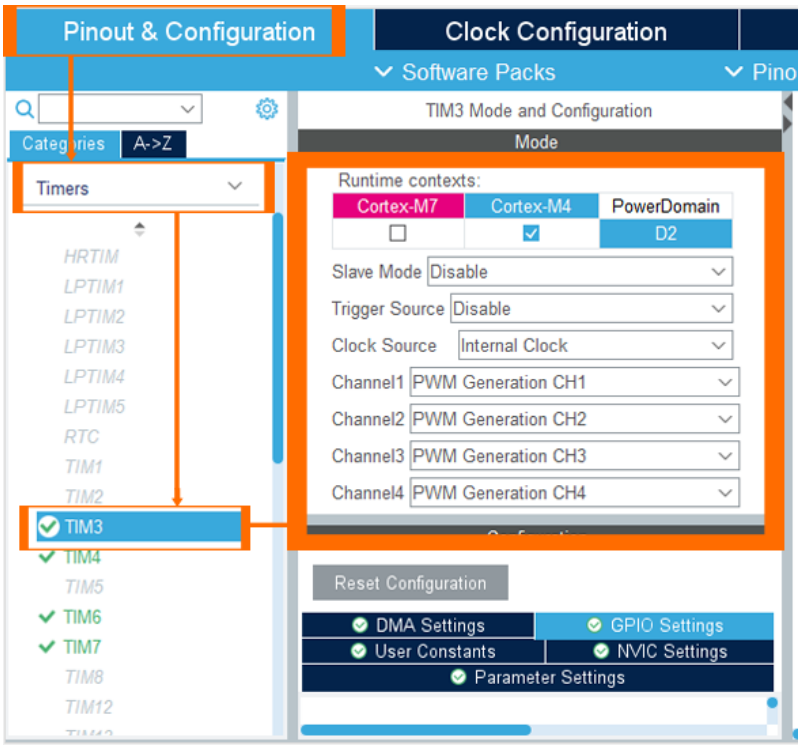


Fig 5.35 settaggio dei canali in modalità “PWM Generation”

Per settare la frequenza dei segnali PWM si utilizza la seguente formula:

$$Update\ Event = \frac{Timer_{clock}}{(Prescaler+1)(Period+1)}$$

Quindi, per avere una frequenza di 50 Hz:

$$Update\ Event = \frac{64000000}{(6400+1)(200+1)} = 50\ Hz = 20\ ms$$

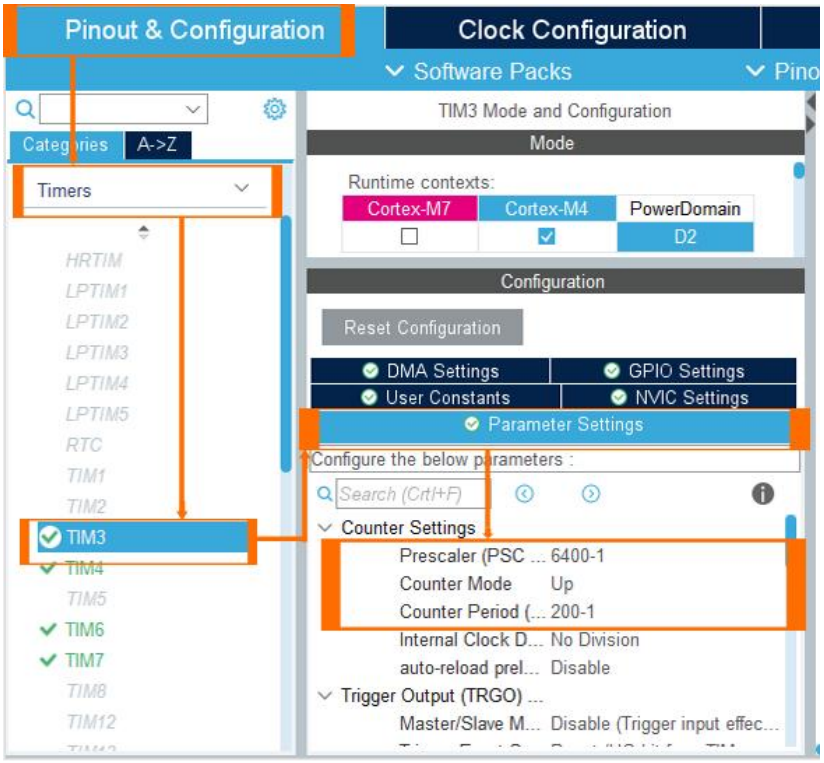


Fig 5.36 settaggio dei parametri per la frequenza dei segnali

Nella sezione “GPIO Settings” vengono specificati automaticamente i pin d’uscita dei segnali PWM.

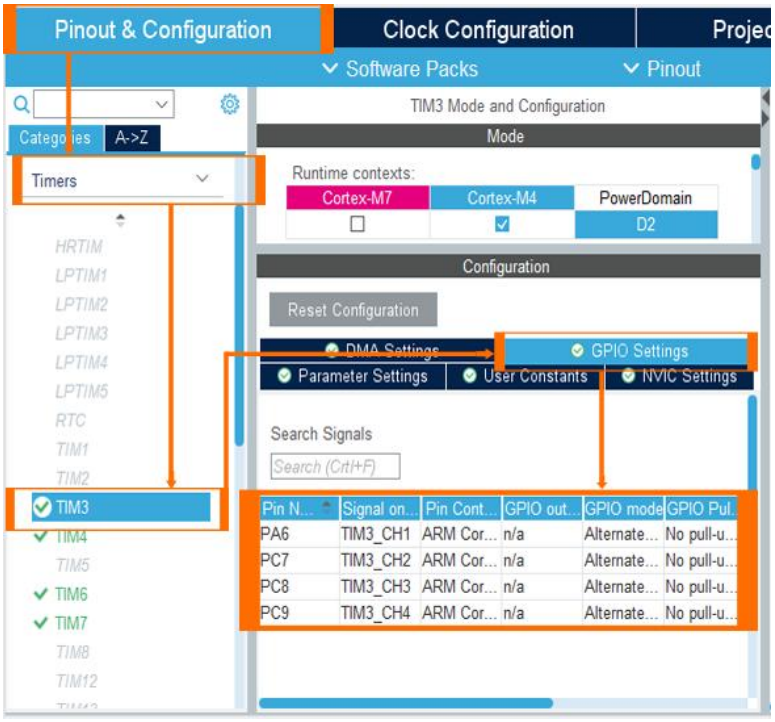


Fig 5.37 i Pin d’uscita dei segnali PWM

Per attivare i canali nella modalità “PWM Generation”, esprimendolo in codice, si utilizza la funzione **HAL_TIM_PWM_Start(TIM_HandleTypeDef *htim, uint32_t Channel)** della libreria HAL.

```
void MotorsArm(TIM_HandleTypeDef *htim,TIM_TypeDef *TIM){

      HAL_TIM_PWM_Start(htim,TIM_CHANNEL_1);
       TIM->CCR1=9.5;
       HAL_TIM_PWM_Start(htim,TIM_CHANNEL_2);
       TIM->CCR2=9.5;
       HAL_TIM_PWM_Start(htim,TIM_CHANNEL_3);
       TIM->CCR3=9.5;
       HAL_TIM_PWM_Start(htim,TIM_CHANNEL_4);
        TIM->CCR4=9.5;

   }
```

Fig 5.38 attivazione canali e armamento motori

Nel codice riportato nella figura 5.30, avviene l’armamento dei motori assegnando all’attributo **TIM->CCRN** il valore 9.5. Questo valore viene calcolato tramite la seguente formula:

$$\frac{CCRN * 100}{Periodo} = 4.75$$

In questo caso il periodo è uguale a 200 (figura 5.28) quindi,

$$CCRN = \frac{200 * 4.75}{100} = 9.5$$

Una volta avvenuto con successo l'armamento, l'ESC emetterà' due BEEP veloci poi Un BEEP lungo. Le velocità ottenute nella figura 5.33 vengono poi trasformate in un range compreso tra 6% e 10% di duty cycle (corrispondono ad un valore di CCRN compreso tra 11 e 20), poiché possano essere percepite dai motori con un duty cycle consono.

Per la spiegazione della funzione **map()** vedi figura 5.26

```
PWM_1 = map(*(Speeds+0), 0, MOTOR_MAX_SPEED_3, MOTOR_MIN_UP, throttle);
PWM_2 = map(*(Speeds+1), 0, MOTOR_MAX_SPEED_3, MOTOR_MIN_UP,throttle);
PWM_3 = map(*(Speeds+2), 0, MOTOR_MAX_SPEED_3, MOTOR_MIN_UP, throttle);
PWM_4 = map(*(Speeds+3), 0, MOTOR_MAX_SPEED_3, MOTOR_MIN_UP, throttle);


TIM3->CCR1=PWM_1; //send pwm signal to motor 1
TIM3->CCR2=PWM_2;//send pwm signal to motor 2
TIM3->CCR3=PWM_3;//send pwm signal to motor 3
TIM3->CCR4=PWM_4;//send pwm signal to motor 4
```

Fig 5.39 Invio segnali PWM ai motori

Una volta trasformate le velocità nel range desiderato, in questo caso 11 come minimo e throttle come massimo che rappresenta la spinta letta dal telecomando, vengono assegnati a CCRN di ogni canale PWM per essere inviati ai motori.

Cambianodo il valore di CCRN, varia il duty cycle del segnale PWM.

IL calcolo dei PID e l'invio dei segnali PWM avvengono con una frequenza di

100 Hz, perciò, è stato attivato il Timer6 in modalità interrupt tramite i seguenti passaggi:

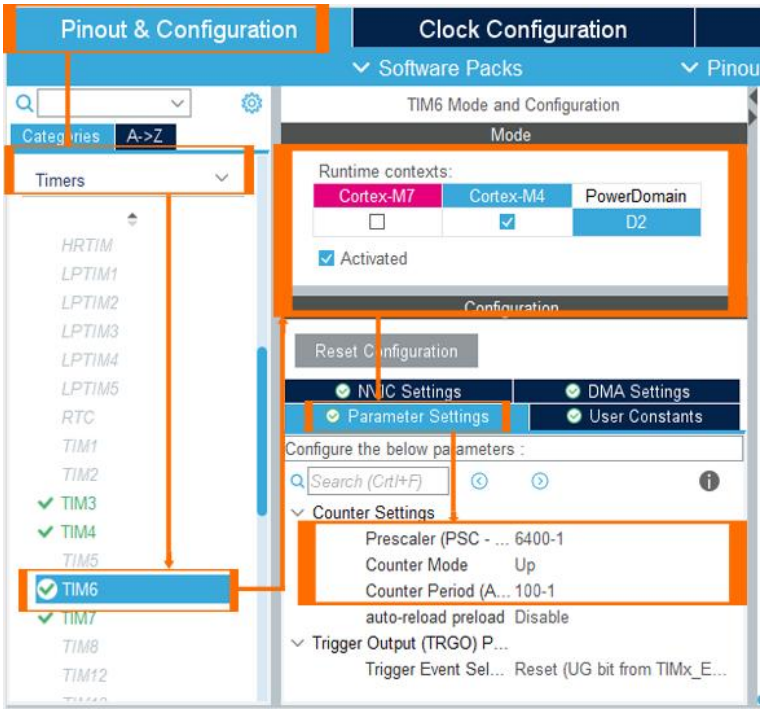


Fig 5.40: Attivazione e settaggio parametri Timer6

Per settare la frequenza del timer si utilizza la solita formula:

$$Update\ Event = \frac{Timer_{clock}}{(Prescaler+1)(Period+1)}$$

Quindi, per avere una frequenza di 250 Hz:

$$Update\ Event = \frac{64000000}{(6400+1)(100+1)} = 100\ Hz = 10\ ms$$

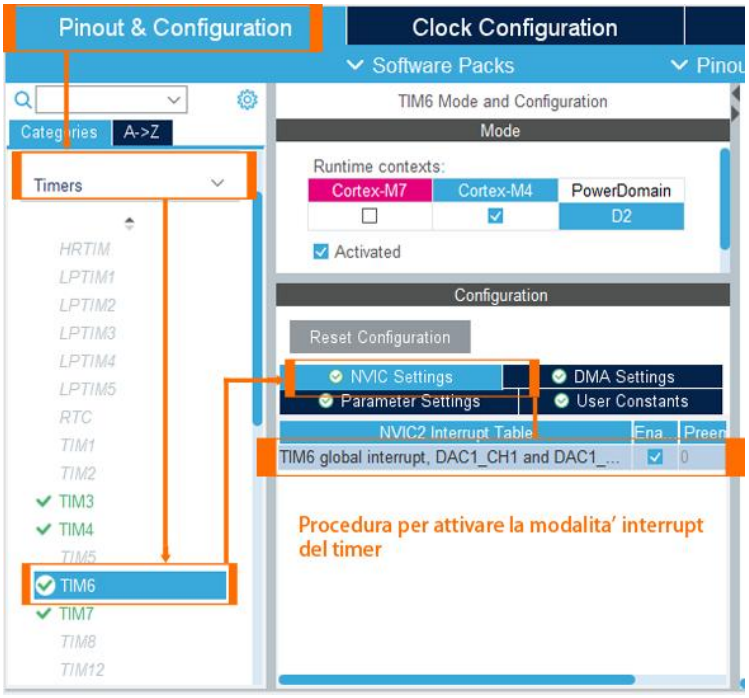


Fig 5.41: Attivazione modalità interrupt Timer6

6 PROVE SPERIMENTALI

In questo capitolo vengono trattate le prove effettuate durante lo sviluppo del quadrirotore. Inoltre, vengono discusse le risoluzioni dei vari errori affrontati.

6.1 OSCILLAZIONI DELL'ASSE X E Y

Nelle prime prove effettuate, il quadrirotore sviluppava sempre delle oscillazioni ad alte frequenze lungo i due assi X e Y. Provando a tarare i valori dei PID, le oscillazioni comunque non si attenuavano. Analizzando la filtrazione dei dati acquisiti dal sensore, è risultato che i parametri del filtro Kalman non erano tarati in maniera ottimale, infatti, durante il funzionamento dei motori i valori acquisiti degli angoli non erano precisi e presentavano forti oscillazioni. Per ovviare a questa problematica, sono stati aumentati i valori della matrice di varianza R e diminuiti quelli dalla matrice di covarianza Q [8].

Di seguito si illustrano i diagrammi di stima degli angoli senza la taratura con l'assetto reale del quadrirotore uguale a zero gradi:

```
Kalman_t KalmanX = {
        .Q_angle = 0.001f,
        .Q_bias = 0.003f,
        .R_measure = 0.03f
};

Kalman_t KalmanY = {
        .Q_angle = 0.001f,
        .Q_bias = 0.003f,
        .R_measure =0.03f,
};
```

Fig 6.1: valori delle matrici R e Q

Fig 6.2: diagramma stima angolo pitch con motori fermi (angolo reale zero gradi)

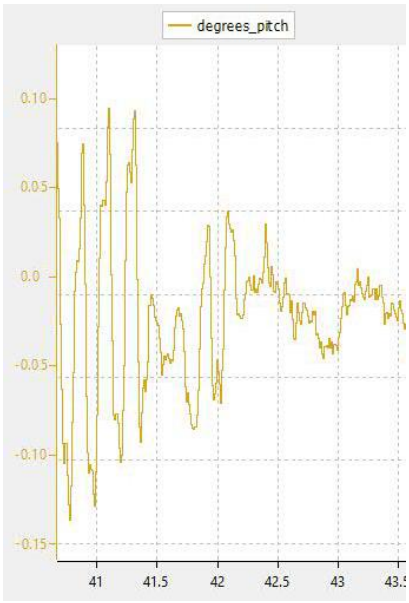


Fig 6.3: diagramma stima angolo pitch con motori in rotazione senza le eliche (angolo reale zero gradi)

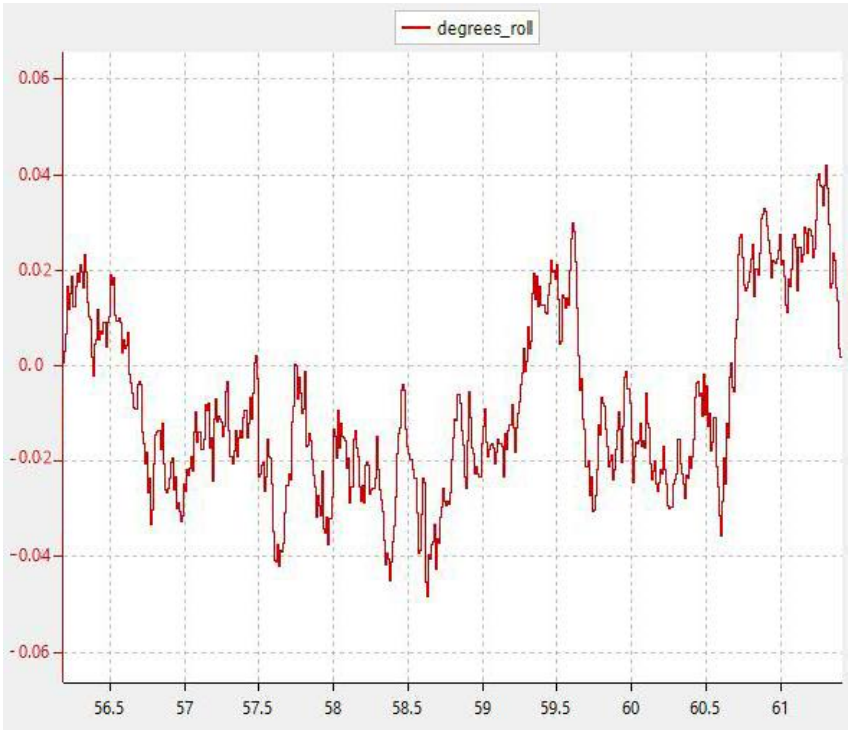


Fig 6.4: diagramma stima angolo roll con motori fermi (angolo reale zero gradi)

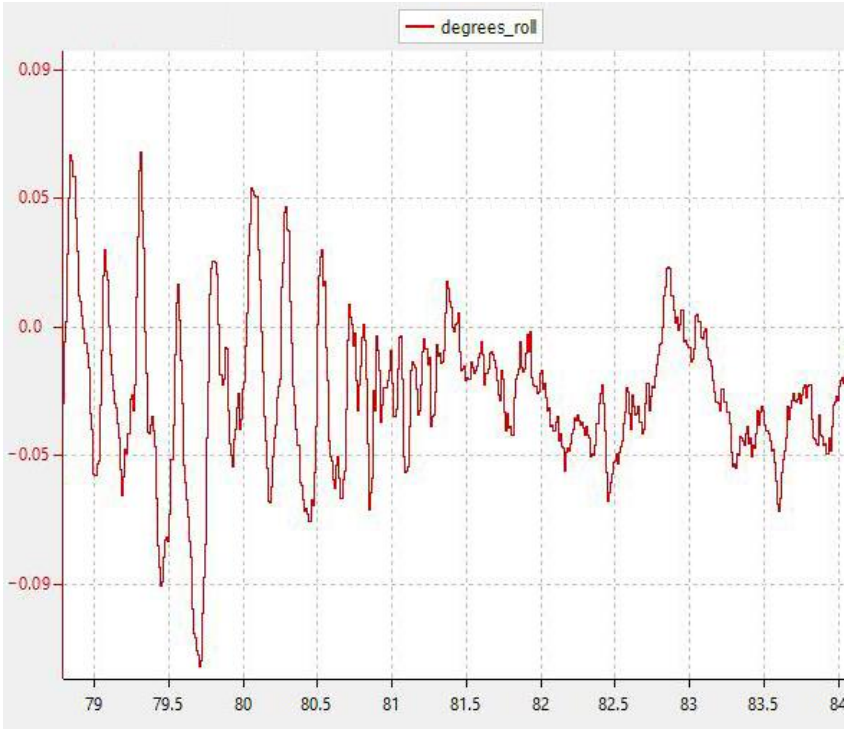


Fig 6.5: diagramma stima angolo roll con motori in rotazione senza le eliche (angolo reale zero gradi)

Di seguito invece si illustrano i diagrammi di stima degli angoli dopo la taratura delle matrici:

```
Kalman_t KalmanX = {
        .Q_angle = 0.1f,
        .Q_bias = 0.05f,
        .R_measure = 715.0f
};

Kalman_t KalmanY = {
        .Q_angle = 0.1f,
        .Q_bias = 0.05f,
        .R_measure =715.0f,
};
```

Fig 6.6: valori delle matrici R e Q tarati

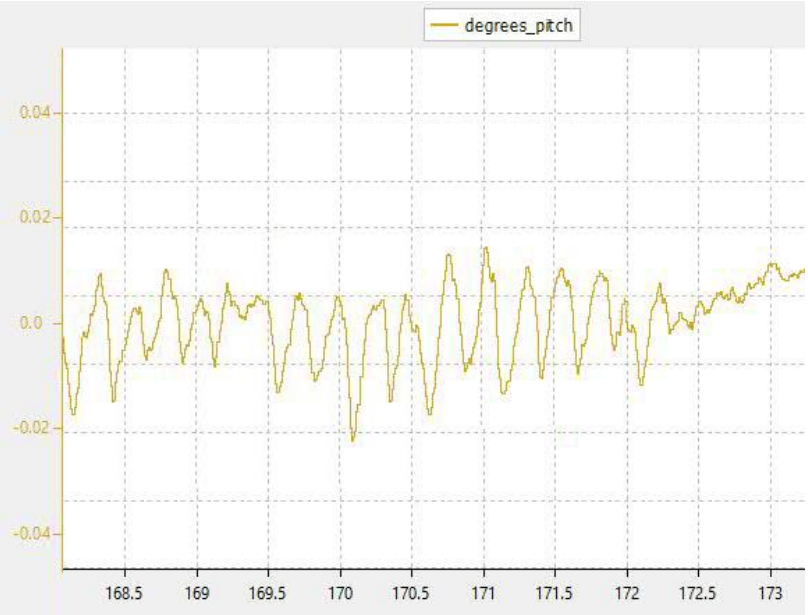


Fig 6.7: diagramma stima angolo pitch con motori in rotazione senza eliche (angolo reale zero gradi)

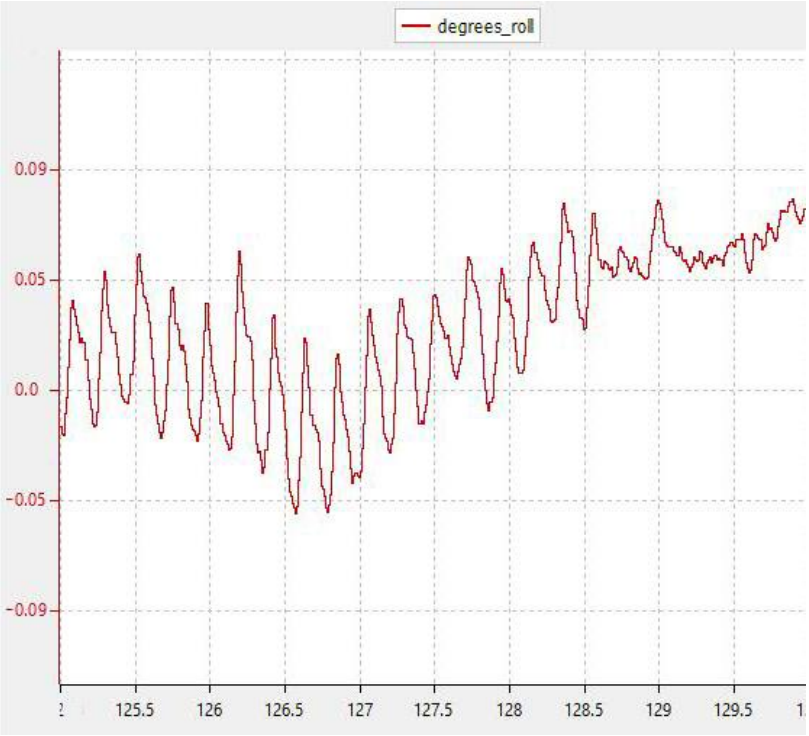


Fig 6.8: diagramma stima angolo roll con motori in rotazione senza le eliche (angolo reale zero gradi)

Nei diagrammi 6.6 e 6.7 si noti che, le oscillazioni durante il funzionamento dei motori sono molte ridotte rispetto a quelle nei diagrammi 6.3 e 6.5.

6.2 ROTAZIONE ATTORNO ALL'ASSE Z

Dopo aver risolto la precedente problematica, il quadrirotore ha cominciato a ruotare attorno al suo asse z in maniera incontrollabile. Cercando di risolvere questa anomalia, è stato scoperto che questo comportamento del quadrirotore può derivare da tre principali errori:

1- montatura errata delle eliche;
2- calibrazione offset dell'asse Z del giroscopio;
3- calibrazione delle ESC.

Esaminando il quadrirotore, le eliche erano montate in maniera corretta, mentre l'asse Z del giroscopio non era calibrato. Infatti, aveva un errore di offset pari a

-0.03 gradi/sec che, è stato risolto sommando, ad ogni lettura dal sensore, 0.03 gradi/sec per portarlo a zero.

L'asse Z del giroscopio è stato calibrato prima tramite la funzione **calibrate_MPU_6050**(vedi figura 5.17) però è servita un ulteriore calibrazione di offset per eliminare questo errore.

Infine, per risolvere completamente questo errore è servita anche la calibrazione dell'ESC.

Per calibrare le ESC, che praticamente si tratta di una procedura per definire il range minimo e massimo del duty cycle, si effettuano i seguenti passaggi:

1- si invia un segnale PWM del massimo duty cycle desiderato (in questo caso 10%);

2- si connette la batteria all'ESC poi si aspettano 2 secondi;

3- l'ESC emetterà due BEEP veloci per segnalare che il massimo duty cycle è stato confermato;

4- si invia un segnale PWM del minimo duty cycle desiderato (in questo caso 6%)
5- L'ESC emetterà due BEEP lunghi per segnalare che il minimo duty cycle è stato confermato.

6.3 TARATURA CONTROLLORE PID

La taratura del controllore PID consiste nel determinare il guadagno del termine proporzionale, derivativo e integrale. Innanzitutto, prima di mostrare il metodo di taratura, si specifica che non tutti gli errori derivano dai PID. Infatti, non tutte le vibrazioni sono causate da valori alti di K_P o K_D. Si deve eliminare tutte le fonti di vibrazione prima procedere con la taratura. I motori e le eliche dovrebbero essere bilanciati, il sensore dovrebbe essere montato in modo da rimuovere possibili vibrazioni e non interferire con le letture dati del giroscopio. Idealmente, il baricentro dovrebbe trovarsi esattamente al centro di massa del quadrirotore. Quando questa situazione non si verifica, alcuni motori dovranno spingere di più per controbilanciare il velivolo. Questo causerà il surriscaldamento dei motori, limiterà la velocità massima e influenzerà la stabilità dell'intero quad.

Inizialmente, per tarare i PID, è stato aumentato il valore K_P gradualmente, finché il quadrirotore ha cominciato ad oscillare con alte frequenze. Dopodiché K_P è stato diminuito del 50%. Di conseguenza, si è cominciato ad aumentare il valore K_D , fino al punto in cui si sono verificati degli scatti da parte dei motori. Per attenuare questo comportamento è stato diminuito il valore K_D . Infine, per il valore K_i è stata applicata la stessa procedura di prima. È stato aumentato K_i finche' il quadrirotore ha sviluppato delle oscillazioni a basse frequenze, di conseguenza è stato diminuito del 50%.

Ricapitolando, il valore di guadagno K_P è uguale a 50% del valore al quale inizia a oscillare il quadrirotore, mentre il guadagno K_D è un valore vicino e minore del valore al quale iniziano a scattare i motori. Infine, K_i è uguale a 50% del valore al quale il quadrirotore inizia ad oscillare a basse frequenze.

6.4 PROVE DEL TELECOMANDO

Tutte le prove illustrate precedentemente sono state effettuate con l'uso del telecomando, è stato utile dal punto di vista della sicurezza, in quanto si riusciva a telecomandare il quadrirotore da lontano.

Nelle prime prove sono stati disattivati i controlli degli angoli Pitch, Roll e Yaw (cioè l'angolo desiderato era settato a zero gradi) per diminuire il cerchio di ricerca di eventuali errori.

Invece, nelle ultime prove sono stati attivati i controlli degli angoli, avendo il quadrirotore fissato dai bracci.Una volta tarati i PID, il quadrirotore rispondeva in maniera abbastanza veloce alle variazioni degli angoli desiderati inviati tramite il telecomando ma con leggere oscillazioni che saranno approfondite nel prossimo capitolo.

6.5 RISULTATI FINALI

Il quadrirotore è riuscito a volare come previsto, rimanendo in quota, ma presenta alcune criticità.

Nelle prove definitive, infatti, sono state riscontrate le problematiche seguenti: leggere oscillazioni ad alta frequenza lungo l'asse X ed Y e deviazione orizzontale in direzione positiva dell'asse X, che causano qualche instabilità nella fase di volo.

Ipotizzo che tali comportamenti evidenziati dal quadrirotore in fase di volo possano derivare da un malfunzionamento nell'hardware, quindi delle ESC, dei motori o delle eliche. Lo si può verificare scambiando tra loro i motori o le eliche, in modo da rendersi conto quale di questi risulti difettoso.

7 CONCLUSIONE

Affrontare una progettazione di questo tipo non è mai facile, ma, attraverso studi, ricerche approfondite, prove pratiche e soprattutto passione per tale attività, è stato possibile raggiungere un livello di conoscenze tale da permettermi anche di accrescere notevolmente l'autostima oltre ogni aspettativa.

Mi sono particolarmente appassionato alla progettazione di questo velivolo e alla sua realizzazione pratica e tutto questo processo, fatto di fasi ideative, momenti teorici e concreti, mi ha talmente entusiasmato da farmi desiderare ulteriori sperimentazioni future sempre più complesse.

8 SVILUPPI FUTURI

Di seguito si illustrano alcuni sviluppi futuri per migliorare la stabilità e le funzionalità del quadrirotore:

• utilizzare il filtro Madgwick che basato sui quaternioni per evitare i punti di singolarità;

• aggiungere un sensore di distanza per stimare l'altezza del quadrirotore rispetto al suolo. Con questo sensore si può sviluppare un meccanismo di decollo automatizzato;

• aggiungere un magnetometro per stimare meglio la velocità angolare dell'asse Z;

• aggiungere un dispositivo GPS per sviluppare algoritmi di voli autonomi;

• aggiungere una telecamera per sviluppare algoritmi di tracciamento oggetti utilizzando l'intelligenza artificiale.

Bibliografia

[1] Andrea Bonci. UNMANNED AERIAL VEHICLES DYNAMICS, Slide del corso magistrale di Dynamics and Control of Intelligent Robots and Vehicles.

[2] Kenzo Nonami •Farid Kendoul • Satoshi Suzuki Wei Wang • Daisuke Nakazawa: Autonomous Flying Robots, Unmanned Aerial Vehicles and Micro Aerial Vehicles.

[3] Pushpendra Kumar : Articolo Estimation of the thrust coefficient of a Quadcopter Propeller using Computational Fluid Dynamics.

[4] Norwegian University of Science and Technology. Andreas Vikane Hystad. Model, Design and Control of a Quadcopter: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2352467/12758_FULLTEXT.pdf?sequence=1&isAllowed=y.

[5] Univesità Federico II, Napoli, Facoltà di Ingegneria. Relazione Controllo di un Quadcopter: http://wpage.unina.it/framato/materiale%20didattico/Tesine/quadricottero/Tesina_Quadricottero.pdf.

[6] California State University, Sacramento.Department of Electrical and Electronic Engineering. Naveen Prabu Palanisamy. Relazione FILTERING OF IMU DATA USING KALMAN FILTER: https://scholarworks.calstate.edu/downloads/dv13zt241.

[7] A practical approach to Kalman filter and how to implement it: http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/.

[8] Val Smirnov,Making Android controlled Arduino quadcopter from scratch: https://www.linkedin.com/pulse/making-android-controlled-arduino-quadcopter-from-scratch-val-smirnov.

[9] Betaflight. Flight controller open source: https://github.com/betaflight/betaflight.

[10] Comunicazione I2C: https://sites.google.com/site/sitodelguru/elettronica/il-protocollo-i2c.

[11] Stm32_45ZI_Q datasheet: https://www.st.com/en/evaluation-tools/nucleo-h745zi-q.html.

[12] Telaio datasheet:

https://dl.djicdn.com/downloads/flamewheel/en/F450_User_Manual_v2.2_en.pdf.

[13] Motori DX2205 datasheet :https://www.amazon.it/DroneAcc-DX2205-2300KV-Brushless-Racing/dp/B075731ZJM/ref=sr_1_1?__mk_it_IT=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=PL5QBUACI7V9&keywords=DX2205&qid=1670435340&sprefix=dx2205%2Caps%2C135&sr=8-1.

[14] Eliche 5045 datasheet: https://www.drone24hours.com/product/dalprop-t5045c-pro/?lang=en.

[15] MPU_6050 datasheet: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[16] Batteria Ovonic Air datasheet: https://www.ovonicshop.com/products/ovonic-11-1v-3s-3000mah-50c-lipo-battery-pack

[17] Telecomando Microzone 6CH datasheet: https://www.katranji.com/tocimages/files/390729-235499.pdf.

[18] Andrea Bonci. Slide del corso triennale di laboratorio di automazione.

[19] PWM Input capture mode : https://www.st.com/content/ccc/resource/training/technical/product_training/c4/1b/56/83/3a/a1/47/64/STM32L4_WDG_TIMERS_GPTIM.pdf/files/STM32L4_WDG_TIMERS_GPTIM.pdf/jcr:content/translations/en.STM32L4_WDG_TIMERS_GPTIM.pdf.

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE