



**Politecnico  
di Torino**

Ingegneria aerospaziale

A.a. 2023/2024

Sessione di Laurea dicembre 2024

# **Configurazione e sviluppo software dell'autopilota di un quadricottero e analisi di prestazione dei sistemi di localizzazione di precisione**

Relatore:

dr. Stefano PRIMATESTA

Candidato:

Pietro BELTRAMI

Correlatore:

ing. Riccardo ENRICO

# Abstract

I Sistemi Aeromobili a Pilotaggio Remoto (SAPR), anche chiamati UAS (Unmanned Aircraft Systems) o, più semplicemente, droni rappresentano una delle frontiere più avanzate ed innovative nell'ambito del trasporto e dei servizi aerei. I progressi della ricerca e dello sviluppo tecnologico hanno permesso, negli ultimi decenni, di espandere notevolmente il campo di applicazione di questi veicoli e molti sono gli scenari di missione in cui, con buona confidenza, verranno adoperati in futuro.

Tra le configurazioni di UAS allo stato dell'arte, i quadricotteri rappresentano la più comune, soprattutto grazie alla disponibilità sul mercato di un'ampia gamma di veicoli di questo tipo, adatti a soddisfare diverse esigenze. In particolare, le piattaforme più apprezzate in ambito di ricerca e sviluppo di UAS, sono quelle che permettono una versatilità sia hardware che software, con la possibilità di aggiungere o sostituire componenti e modificare il software di bordo. Tuttavia, per sfruttare un veicolo versatile al massimo delle sue potenzialità, è necessario un intervento di configurazione per ottimizzarne le performance e per adeguare le funzioni che esso può svolgere sulla base delle necessità dell'utente.

Il lavoro svolto nell'ambito di questa tesi ha l'obiettivo di rispondere a questa esigenza, configurando in modo ottimale il drone PX4 Vision Autonomy Development Kit in dotazione al Gruppo di ricerca, accertandosi di possedere una piena conoscenza dei componenti del veicolo e di come questi interagiscano. Questo perché il drone attenzionato non è più ufficialmente supportato dalla comunità e, vista l'esigenza di operare con le più recenti versioni del software di bordo, esso necessita di essere manualmente aggiornato. Alla base vi è quindi lo studio del funzionamento del drone e delle sue parti, cui segue il lavoro di aggiornamento e configurazione del software installato su entrambi i computer di bordo.

In una fase successiva il velivolo configurato sarà quindi sfruttato per effettuare una campagna di sperimentazione: il gruppo di ricerca dispone di diversi supporti tecnologici per la determinazione della posizione del veicolo mediante GPS aumentato. È necessario che i vari metodi di correzione del segnale satellitare siano valutati, allo scopo di stilare un benchmark che veda i vari equipaggiamenti ordinati secondo la massima precisione raggiungibile. La trattazione si concluderà quindi con la valutazione dei risultati della campagna sperimentale.

# Indice

<b>Abstract</b> .....	<b>2</b>
<b>Indice</b> .....	<b>3</b>
<b>Lista delle figure</b> .....	<b>6</b>
<b>Introduzione</b> .....	<b>8</b>
Background.....	9
Obiettivo .....	11
Sommaro .....	12
<b>1) Veicolo e componenti hardware</b> .....	<b>13</b>
<b>1.1 Veicolo</b> .....	<b>13</b>
<b>1.2 Contenuto del kit</b> .....	<b>14</b>
<b>1.3 Setup per il volo</b> .....	<b>18</b>
<b>1.4 Hardware e cablaggi</b> .....	<b>19</b>
1.4.1 Flight controller .....	20
1.4.2 Power management board .....	23
1.4.3 Companion computer .....	25
1.4.4 Modulo GNSS .....	25
1.4.5 Camera .....	28
1.4.6 Optical flow e distance sensor .....	30
1.4.7 Cablaggi .....	31
<b>2) Autopilota e framework setup</b> .....	<b>34</b>
<b>2.1 Autopilota</b> .....	<b>34</b>

2.1.1 Software .....	36
2.1.2 Flight stack .....	38
2.1.3 Middleware .....	39
2.1.4 Sistema operativo .....	39
<b>2.2</b> Interfaccia e gestione dell'autopilota mediante ground station .....	<b>40</b>
<b>2.3</b> Framework setup .....	<b>42</b>
2.3.1 ROS .....	43
2.3.2 Sistema operativo .....	47
2.3.3 MAVROS, MAVLink e $\mu$ XRCE-DDS .....	48
2.3.4 Camera .....	51
<b>3) Setup sperimentale .....</b>	<b>54</b>
<b>3.1</b> Modifiche hardware.....	<b>54</b>
3.1.1 Modulo GNSS .....	54
3.1.2 Camera .....	56
<b>3.2</b> Determinazione della posizione .....	<b>58</b>
3.2.1 Autopilota PX4 – filtro di Kalman esteso .....	58
3.2.2 Sistemi GNSS .....	61
3.2.3 RTK.....	63
3.2.4 Spin3 GNSS .....	65
3.2.5 ArUco marker .....	67
3.2.6 Setup .....	71
<b>4) Risultati .....</b>	<b>73</b>
<b>4.1</b> Validazione della configurazione.....	<b>73</b>
<b>4.2</b> Test dei sistemi per la determinazione della posizione .....	<b>76</b>

4.2.1 Modulo GNSS e ArUco marker .....	77
4.2.2 Modulo GNSS, ArUco marker e antenna RTK .....	82
4.2.3 Modulo GNSS, ArUco marker e correzioni Spin3 .....	87
4.2.4 Confronti finali .....	91
<b>5) Conclusioni .....</b>	<b>95</b>
<b>Future work .....</b>	<b>97</b>
<b>Ringraziamenti.....</b>	<b>98</b>
<b>References.....</b>	<b>99</b>

# Lista delle figure

Figura 1: Drone PX4 Vision Autonomy Development Kit [4].....	13
Figura 2: Contenuto del Kit [4] .....	14
Figura 3: Vista dall'alto del drone e dei suoi computer smontati [2] .....	17
Figura 4: Vista esplosa dei componenti principali del drone [4].....	19
Figura 5: Flight controller Pixhawk 6C [5].....	20
Figura 6: Slot per scheda SD [7] .....	22
Figura 7: Power management board [9].....	23
Figura 8: Schema esplicativo della funzione del power module [8].....	23
Figura 9: Schema esplicativo della funzione della power distribution board [8] .....	24
Figura 10: Modulo GNSS e connessione con il flight controller [12].....	26
Figura 11: Schema del modulo GNSS H-RTK M8N [34].....	27
Figura 12: Collegamento dei moduli GNSS primario e secondario con il flight controller [8] .....	28
Figura 13: Camera Occipital Structure Core [15] .....	28
Figura 14: Test vista RGB e di profondità della camera .....	29
Figura 15: Sensore optical flow [17] .....	30
Figura 16: Distance sensor [18] .....	30
Figura 17: Schema dei cablaggi [19] .....	32
Figura 18: Architettura dell'autopilota con companion computer [20].....	35
Figura 19: Schema dell'architettura di alto livello del software di PX4 [22].....	37
Figura 20: Schema a blocchi del flight stack [22] .....	38
Figura 21: Schermata principale di QGroundControl .....	40
Figura 22: Schermata di aggiornamento del firmware in QGroundControl .....	41
Figura 23: Schema di funzionamento del middleware $\mu$ XRCE-DDS con PX4 e ROS 2 [32] .....	50
Figura 24: Avvio del client $\mu$ XRCE-DDS dal companion computer.....	51
Figura 25: Modulo GNSS F9P installato sul veicolo in sostituzione del M8P [35].....	55
Figura 26: RealSense Depth Camera D435i da aggiungere al veicolo [36].....	57
Figura 27: Icona e informazioni sul GPS in QGroundControl.....	62
Figura 28: Base RTK F9P [35].....	64
Figura 29: Dati RTK su QGroundControl .....	65
Figura 30: Stazioni della rete Spin3 [45].....	66
Figura 31: GPS lock RTK con Spin3, visualizzazione dei dati da QGroundControl.....	67
Figura 32: Esempi di ArUco marker [47] .....	68
Figura 33: ArUco marker (id 00) con sistema di riferimento [48] .....	69
Figura 34: Schema dei nodi e topic ROS per camera e ArUco detection.....	70

Figura 35: Visualizzazione esemplificativa dei dati pubblicati sul topic /aruco_ detections .....	71
Figura 36: Spostamenti nel piano x-y .....	74
Figura 37: Spostamento longitudinale nel tempo .....	74
Figura 38: Spostamento laterale nel tempo .....	74
Figura 39: Spostamento verticale secondo le misurazioni del distance sensor .....	75
Figura 40: Valori assunti dall'angolo di yaw durante il volo .....	75
Figura 41: Comandi impartiti durante il volo.....	76
Figura 42: Sistemi di riferimento a confronto [1] [48].....	77
Figura 43: Volo con GPS – Spostamento longitudinale.....	78
Figura 44: Volo con GPS – Spostamento laterale .....	78
Figura 45: Volo con GPS – Spostamento verticale.....	79
Figura 46: Confronto tra posizione locale e globale.....	79
Figura 47: Volo con GPS – Spostamenti nel piano (marker).....	80
Figura 48: Volo con GPS – Spostamenti nel piano (dati GPS e global position).....	80
Figura 49: Volo con GPS – effetto del sensor fusion.....	81
Figura 50: Volo con GPS – prestazioni del GPS.....	82
Figura 51: Volo con RTK – prestazioni del GPS .....	83
Figura 52: Volo con RTK – Spostamenti nel piano (dati GPS e global position) .....	83
Figura 53: Volo con RTK – effetto del sensor fusion.....	84
Figura 54: Volo con RTK – Spostamento longitudinale.....	85
Figura 55: Volo con RTK – Spostamento laterale .....	85
Figura 56: Volo con RTK – Spostamento verticale.....	86
Figura 57: Volo con RTK – Spostamenti nel piano (marker) .....	86
Figura 58: Volo con Spin3 – prestazioni del GPS.....	87
Figura 59: Volo con Spin3 – Spostamenti nel piano (dati GPS e global position) .....	87
Figura 60: Volo con Spin3 – effetto del sensor fusion.....	88
Figura 61: Volo con Spin3 – Spostamento longitudinale .....	89
Figura 62: Volo con Spin3 – Spostamento laterale .....	89
Figura 63: Volo con Spin3 – Spostamento verticale.....	90
Figura 64: Volo con Spin3 – Spostamenti nel piano (marker).....	90
Figura 65: Instabilità del GPS fix con Spin3.....	91
Figura 66: Errore di posizione lungo le direzioni del piano durante i tre test.....	92
Figura 67: Volo con GPS – Traiettoria nel piano durante il volo a punto fisso .....	93
Figura 68: Volo con RTK – Traiettoria nel piano durante il volo a punto fisso.....	94
Figura 69: Volo con Spin3 – Traiettoria nel piano durante il volo a punto fisso .....	94

# Introduzione

I cosiddetti UAV (acronimo per unmanned aerial vehicle) rappresentano oggi la categoria di velivoli più avanguardistica: questi veicoli, siano essi a pilotaggio remoto o persino a guida autonoma, continuano ad offrire sempre nuovi sbocchi di utilizzo contribuendo a ridefinire la stessa concezione di trasporto e lavoro aereo. In particolare, i multicotteri sono già da tempo impiegati in diversi scenari di missione con le applicazioni più disparate, quali sorveglianza, ricerca e soccorso, fotografia e riprese aeree ma anche per scopi ricreativi. Inoltre, per sottolineare ulteriormente l'importanza presente e futura di questi particolari velivoli, è necessario aggiungere all'elenco anche i servizi postali o, più in generale, la urban air mobility oltre a numerose altre applicazioni oggi in fase di sperimentazione che sono però considerate particolarmente promettenti. Quello degli UAV è quindi uno dei campi di ricerca più attivi in ambito aeronautico, proprio perché capace di produrre soluzioni nuove o più facilmente implementabili, anche dal punto di vista economico, e persino con un minore impatto ambientale rispetto all'uso di elicotteri o altri velivoli convenzionali.

Come accade nella maggior parte delle applicazioni ingegneristiche, anche i multicotteri a pilotaggio remoto nascono dall'incontro di tecnologie afferenti a discipline differenti, dalla meccanica del volo alla robotica, ed in particolare si collocano a cavallo tra l'ambito sistemistico aeronautico e quello mecatronico: di fatto, questi veicoli sono costituiti da una serie di componenti hardware il cui funzionamento combinato e correttamente coordinato rappresenta l'aspetto più importante per garantirne la funzionalità e la conseguente capacità di svolgere la missione prevista. In tal senso, a rivestire un ruolo di primaria importanza è il software, che governa l'interfaccia ed il funzionamento dei vari componenti e del veicolo completo: se sui velivoli convenzionali il software sta acquisendo, generazione dopo generazione, un'importanza sempre maggiore, per i veicoli unmanned esso è veramente fondamentale. È quindi chiaro che, se si intende sfruttare un quadricottero a pilotaggio remoto per attività di ricerca, così come per qualsiasi altra attività, è necessario che questo sia dotato di un software aggiornato e adeguatamente configurato per rispondere nel miglior modo possibile alle esigenze di missione.

Per quanto concerne le tecnologie su cui questi velivoli fanno affidamento, tra le più importanti spiccano quelle atte alla determinazione della posizione: la conoscenza dell'esatta collocazione del veicolo nello spazio è qualcosa da cui infatti non si può prescindere, non solo per garantire la buona riuscita della missione, ma ancora di più per questioni di sicurezza.



L'ampia diffusione di tecnologie e ricevitori basate sui sistemi di navigazione satellitari fa di questi la scelta più scontata per determinare la posizione di un UAV sul globo, tuttavia, la precisione che questi sistemi garantiscono per applicazioni civili è spesso insufficiente. Missioni di mappatura per l'agricoltura di precisione o anche voli in formazione, per fare solo qualche esempio, richiedono infatti precisioni centimetriche ad oggi non raggiungibili sfruttando esclusivamente i segnali satellitari. È per superare questa criticità che sono nate tecnologie di supporto alla navigazione satellitare che, calcolando in vari modi delle correzioni da applicare ai segnali provenienti dallo spazio, hanno dato origine al cosiddetto GPS aumentato, la cui precisione, notevolmente superiore a quella del tradizionale GPS, dipende naturalmente dal modo in cui il segnale di correzione viene generato.

Il presente lavoro di tesi intende focalizzarsi sui due importanti aspetti appena descritti, ovvero la configurazione software e la determinazione della posizione mediante varie tecnologie di correzione abbinata ai sistemi satellitari. Come veicolo di riferimento sarà adottato il drone PX4 Vision Autonomy Development Kit, in dotazione al gruppo di ricerca, di cui verrà naturalmente anche fornita una descrizione dal punto di vista della componentistica hardware.

## Background

Gli aeromobili a pilotaggio remoto rappresentano oggi l'ultima frontiera di sviluppo nell'ambito del trasporto aereo. Nati in ambito militare, lo scopo dei primi droni realizzati era quello di sostituire degli equipaggi umani nello svolgimento di missioni altamente rischiose; cominciando con semplici aerostati, passando per i V-1 sviluppati dall'esercito nazista e poi i primi tentativi di pilotaggio di aeroplani a distanza, decade dopo decade, autonomia ed affidabilità di questi particolari velivoli è andata sempre crescendo. Con l'avvento del nuovo millennio il grado di maturazione di questa tecnologia era ormai tale da consentirne l'affermazione come valida alternativa, più economica e meno rischiosa, rispetto all'utilizzo di aeroplani a pilotaggio convenzionale, spianando la strada alla diffusione dei velivoli unmanned ed alla loro adozione da parte di numerosi eserciti del mondo nei primi due decenni del 2000. È più o meno nello stesso periodo che, grazie all'evoluzione tecnologica, iniziano a comparire sul mercato i primi droni a scopo commerciale, per applicazioni civili o semplicemente con fini ricreativi. Prende allora avvio una nuova fase storica, in cui i velivoli senza pilota, e soprattutto i multicotteri, cominciano a diffondersi sempre più rapidamente contribuendo a cambiare ed ampliare lo spettro di possibilità concesse dal trasporto e dal lavoro aereo:

produzioni televisive, servizi postali rapidi in aree remote, mappatura di precisione, fotografia aerea e molti altri servizi hanno oggi un costo più economico ed un più alto grado di fattibilità rispetto al passato, quando questi compiti dovevano essere svolti da elicotteri o da satelliti. A ciò si deve poi il crescente successo commerciale dei droni, successo che funge da motore trainante per lo sviluppo di veicoli sempre più tecnologicamente avanzati, affidabili, autonomi e capaci di destreggiarsi in scenari di missione sempre più complessi.

Per poter controllare efficacemente un veicolo a guida autonoma, garantendone la sicurezza e l'efficacia, è necessario disporre di un sistema preciso ed affidabile per la determinazione della posizione e per la navigazione. A tal proposito, i droni nati in ambito militare possono fare affidamento sul massimo livello di precisione fornito dalle reti GNSS, tecnologia sfortunatamente non disponibile per impieghi civili, se non con prestazioni fortemente degradate. Ecco che allora lo sviluppo di tecnologie alternative, che possano affiancare e migliorare le prestazioni ottenibili con i sistemi satellitari, gioca un ruolo fondamentale per l'implementazione di applicazioni commerciali ad uso civile. Sono quindi oggi numerosi i casi in cui, per ragioni di sicurezza o perché la missione stessa lo richiede, si decide di ricorrere all'impiego del cosiddetto GPS aumentato, rendendo possibile lo sfruttamento dei droni in campi di applicazione altrimenti preclusi.

Anche grazie all'ideazione di nuove soluzioni tecnologiche quindi, i velivoli a guida autonoma si sono notevolmente evoluti negli ultimi decenni ed il loro sfruttamento si sta rivelando prezioso in un crescente numero di campi. Con questa consapevolezza, le aziende che si dedicano alla progettazione e produzione di droni o di loro componenti fanno della versatilità un punto di forza per i propri prodotti più venduti. In virtù di questa caratteristica è allora richiesto all'utente uno sforzo di programmazione e più in generale di configurazione, per poter personalizzare il veicolo ed ottimizzarne le caratteristiche in funzione della missione per cui intende impiegarlo. Appare quindi chiaramente l'importanza di questo compito che di fatto permette di trasformare un oggetto volante in una piattaforma che possa essere sfruttata per un uso commerciale o, come nel caso di questa tesi, in uno strumento a servizio della ricerca.

## Obiettivo

Questa tesi si colloca all'interno di un più ampio quadro composto dalle varie attività di ricerca svolte nell'ambito dei veicoli a guida autonoma. Nello specifico, ad essere attenzionato sarà uno dei quattro identici quadricotteri di cui il gruppo di ricerca è in possesso; al termine e grazie al lavoro svolto nell'ambito di questa tesi, questi veicoli saranno quindi utilizzabili per nuovi progetti futuri.

Lo scopo, quindi, è quello di coadiuvare le attività del gruppo di ricerca tramite il raggiungimento di un triplice obiettivo. In primis è richiesto di esaminare e comprendere nel dettaglio il funzionamento generale del quadricottero, quali sono i componenti che lo costituiscono e come interagiscono dal punto di vista funzionale. Il passo successivo è quindi quello di configurare i computer ed i sensori con cui il veicolo è equipaggiato: a monte di questa attività di tesi, il software di cui il drone è dotato non è aggiornato, non è ottimizzato e non è in grado di soddisfare pienamente le necessità del gruppo di ricerca. Inoltre, il veicolo in questione non è più supportato dalla comunità e si necessita quindi di un importante intervento di aggiornamento manuale, che permetta lo sfruttamento del quadricottero al massimo delle sue potenzialità.

Il terzo ed ultimo obiettivo è quello di sfruttare il drone configurato come supporto per una campagna di test di varie tecnologie per la determinazione della posizione. In questo caso, il gruppo di ricerca dispone di diversi equipaggiamenti che possono essere utilizzati a miglioramento del segnale satellitare, è necessario effettuare un confronto prestazionale che porti a stilare un benchmark in ragione della massima precisione che è possibile raggiungere con ciascuna delle tecnologie a disposizione.

## Sommario

La trattazione contenuta nel presente documento si articola sui seguenti cinque capitoli tematici:

- **Capitolo 1** – Veicolo e componenti hardware: si fornisce una descrizione del quadricottero utilizzato come supporto per quest'attività di tesi, dei suoi componenti e di come questi sono tra loro connessi.
- **Capitolo 2** – Autopilota e framework setup: inizia con una panoramica sull'architettura dell'autopilota ed il suo funzionamento; successivamente viene descritta dal punto di vista logico-concettuale la procedura di configurazione, all'interno del computer di missione, del framework che dovrà supportare le funzioni necessarie allo svolgimento dei test di cui ai capitoli successivi.
- **Capitolo 3** – Setup sperimentale: a valle della configurazione del framework, vengono descritte le ulteriori modifiche hardware e software necessarie della campagna di sperimentazione; poi viene introdotta l'attività sperimentale vera e propria, essa si pone lo scopo di stabilire la precisione di diverse tecnologie per la determinazione della posizione sfruttando i sistemi satellitari con un'eventuale correzione.
- **Capitolo 4** – Risultati: vengono presentati, analizzati e discussi i risultati della campagna sperimentale.
- **Capitolo 5** – Conclusioni: la configurazione del veicolo e la fase di sperimentazione vengono analizzate e rivalutate con l'obiettivo di evidenziarne i successi ma soprattutto i punti critici e di miglioramento.

# 1) Veicolo e componenti hardware

## 1.1 Veicolo<sup>[1]</sup>



**Figura 1:** Drone PX4 Vision Autonomy Development Kit

Il drone PX4 Vision Autonomy Development Kit utilizzato per questa tesi è progettato per essere un supporto robusto e versatile da utilizzare come piattaforma di sviluppo per applicazioni di guida autonoma e funzioni di computer vision. È basato sull'autopilota open source PX4, uno dei più ampiamente utilizzati nel settore dei droni.

Per consentire allo sviluppatore che lo adopera di implementare funzionalità avanzate come la navigazione autonoma, la percezione dell'ambiente in cui il drone si trova ed il riconoscimento visivo di target ed ostacoli, questo kit prevede, già inclusi in dotazione, alcuni componenti hardware essenziali. Inoltre, è prevista la possibilità di integrare sensori aggiuntivi, come ad esempio telecamere e LiDAR, per permettendo una comprensione più profonda dell'ambiente circostante se si desidera sviluppare eventuali funzioni che lo richiedano.

Per quanto riguarda il quadricottero, questo viene fornito già parzialmente preassemblato; il flight controller, il companion computer e i sensori sono già fissati alla struttura e connessi tramite opportuni cablaggi secondo

l'architettura che verrà successivamente descritta. È dal punto di vista software che il kit necessita di essere configurato secondo le preferenze e le esigenze dell'utente: è incluso un driver USB contenente il minimo software indispensabile per permettere al kit di volare, ma l'utilizzo di questo dispositivo è fortemente limitante, sia perché il software che contiene non è aggiornato, ma anche e soprattutto perché effettuare il booting del companion computer da un driver esterno dilata inevitabilmente i tempi di processamento.

L'autopilota è gestito dal flight controller che può essere di tipo Pixhawk 4 or Pixhawk 6C a seconda della versione del kit; in questo caso il kit adoperato è della versione V1.5 equipaggiata con il flight controller Pixhawk 6C. Alcune funzioni software sono gestite dal companion computer di tipo Intel UP Core, quali l'interfaccia con la camera di profondità Occipital Structure Core.

## 1.2 Contenuto del kit<sup>[2]</sup>



**Figura 2:** Contenuto del Kit

Con l'acquisto del PX4 Vision Autonomy Development Kit, l'utente entra in possesso dei principali componenti hardware necessari al funzionamento del drone, alcuni ricambi ed alcuni cablaggi e dispositivi di fissaggio aggiuntivi per

facilitare eventuali modifiche o aggiunte hardware. Come verrà a breve specificato, il kit non include tutto il necessario al volo, in ogni caso, vengono forniti i seguenti componenti:

- Core Components:
  - 1x Pixhawk 4 or Pixhawk 6C (for v1.5) flight controller
  - 1x PMW3901 sensore optical flow
  - 1x TOF distance sensor a infrarossi (PSK-CM8JL65-CC5)
  - 1x Structure Core depth camera
    - Vision camera con campo visivo di 160°
    - Camere stereo a infrarossi
    - Onboard IMU
    - Potente processore Multi-core depth NU3000
  - 1x *UP Core* computer (4GB di memoria & 64GB eMMC)
    - Intel® Atom™ x5-z8350 (up to 1.92 GHz)
    - Sistemi operativi compatibili: Microsoft Windows 10 full version, Linux (ubinux, Ubuntu, Yocto), Android
    - FTDI UART connesso al flight controller
    - USB1: USB3.0 porta utilizzata per l'esecuzione di PX4 avoidance environment da chiavetta USB2.0 (la connessione di una periferica USB3.0 può disturbare il GPS).
    - USB2: porta USB2.0 con connettore JST-GH. Può essere utilizzata per una seconda camera, LTE, etc. (or tastiera/mouse in fase di sviluppo).
    - USB3: USB2.0 con porta JST-GH connessa alla depth camera
    - HDMI: HDMI out
    - SD card slot
    - WiFi 802.11 b/g/n @ 2.4 GHz (connesso all'antenna esterna #1). Permette al computer di accedere alle reti WiFi per connettersi a Internet o per gli aggiornamenti.
- Specifiche meccaniche:
  - Telaio: Full 5mm 3k twill in fibra di carbonio
  - Motori: T-MOTOR KV1750
  - ESC: BEHEli-S 20A ESC
  - GPS: M8N GPS module
  - Power module: Holybro PM07

- Larghezza alla base: 286mm
- Massa: 854 grammi senza batteria né eliche
- Telemetria: ESP8266 connessa al flight controller (tramite l'antenna esterna #2). Permette la connessione wireless con la ground station.
- Una chiavetta USB2.0 contenente il software per eseguire:
  - Ubuntu 18.04 LTS
  - ROS Melodic
  - Occipital Structure Core ROS driver
  - MAVROS
  - [PX4 Avoidance](#)
- Cavi assortiti, 8x eliche, 2x battery straps (installati) e altri accessori (che possono essere utilizzati per connettere periferiche aggiuntive).

Come già anticipato, i due computer di bordo (Pixhawk 6C e *UP Core* computer) sono responsabili rispettivamente di supportare l'autopilota e le altre funzioni software. In particolare, i sensori optical flow e Infrared distance sensor sono collegati al computer dell'autopilota, mentre al companion computer è connessa la Structure Core depth camera (si veda a tal proposito lo schema dei cablaggi e relativo paragrafo descrittivo).

È possibile poi notare le numerose possibilità di interfaccia offerte dal companion computer: questo dipende dalla necessità di connettere molteplici altri elementi del drone ma soprattutto per gestire le periferiche, quali monitor e dispositivi di input, necessarie all'utente per interfacciarsi ed eseguire con facilità eventuali operazioni su questo computer.

In origine sul companion computer non è installato alcun elemento software ed è quindi necessario utilizzare la chiavetta USB fornita in dotazione con il kit contenente il sistema operativo (Ubuntu 18.04 LTS) e le altre componenti software necessarie al funzionamento del companion computer, della camera, dell'obstacle avoidance\* e l'interfaccia con l'autopilota.

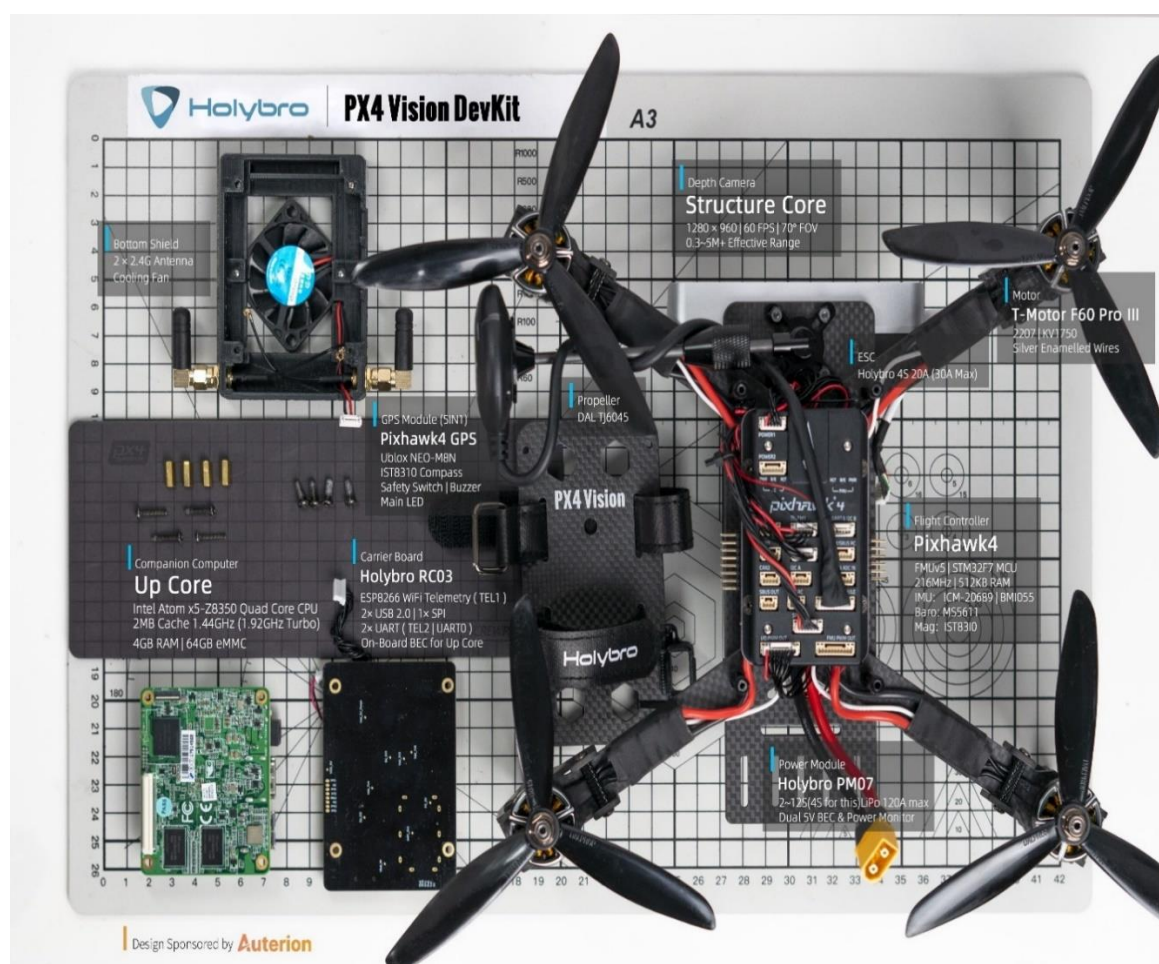
---

\*Nota: sebbene sulla chiavetta sia fornito un esempio funzionante di obstacle avoidance, questo è da ritenersi esclusivamente dimostrativo, dato che questa funzione al momento non è più supportata dall'autopilota; si veda a tal proposito la documentazione ufficiale. <sup>[3]</sup>



La Structure Core depth camera è in realtà costituita da un insieme di più sensori e un'unità di processamento; in particolare le due camere stereo a infrarosso permettono, attraverso la combinazione delle due immagini, di ottenere una ricostruzione tridimensionale dell'ambiente esterno per consentire l'implementazione di applicazioni quali l'obstacle avoidance.

Naturalmente, per migliorare e personalizzare le prestazioni del quadricottero e soprattutto del suo companion computer è opportuno installare il software necessario direttamente a bordo, facendo a meno della chiavetta USB; questa viene infatti fornita in modo che il kit possa essere immediatamente pronto al volo, ma difficilmente soddisfa le esigenze prestazionali e lo specifico grado di personalizzazione e sviluppo che l'utente intende infine raggiungere. Su questo argomento è incentrato il paragrafo 2.3 relativo al framework setup.



**Figura 3:** Vista dall'alto del drone e dei suoi computer smontati

Alcuni elementi indispensabili al volo non sono tuttavia inclusi nel kit: in particolare è necessario acquistare separatamente la batteria che deve essere di tipo 4S LiPo con connettore femmina XT60 e lunga meno di 115 mm per non interferire con il supporto del sensore GPS. Anche il radiocomando non è incluso

e deve essere naturalmente di tipo compatibile con l'autopilota px4; la comunicazione con il drone è permessa da un trasmettitore *FrSky Taranis* con ricevitore R-XSR, anch'esso da acquisire autonomamente, che deve essere installato e connesso con l'autopilota. Per la gestione, configurazione e calibrazione del veicolo, delle sue componenti e dell'autopilota è infine necessario un computer con funzione di ground station su cui deve essere installato il software QGroundControl; si veda a tal proposito la sezione 2.2 relativa proprio a questo argomento.

### 1.3 Setup per il volo <sup>[4]</sup>

Per rendere il kit abile al volo è necessario eseguire alcune operazioni di configurazione e setup degli elementi hardware, principalmente con lo scopo di terminare l'assemblaggio delle parti fornite e soprattutto l'integrazione di quelle non fornite.

A tale scopo è necessario innanzitutto installare il ricevitore per il radiocomando, questo deve essere montato in una posizione tale per cui non vi siano ostacoli ad ostruirne la visuale e deve essere connesso con il flight controller. Successivamente l'antenna GPS deve essere posizionata correttamente sfruttando l'apposito supporto.

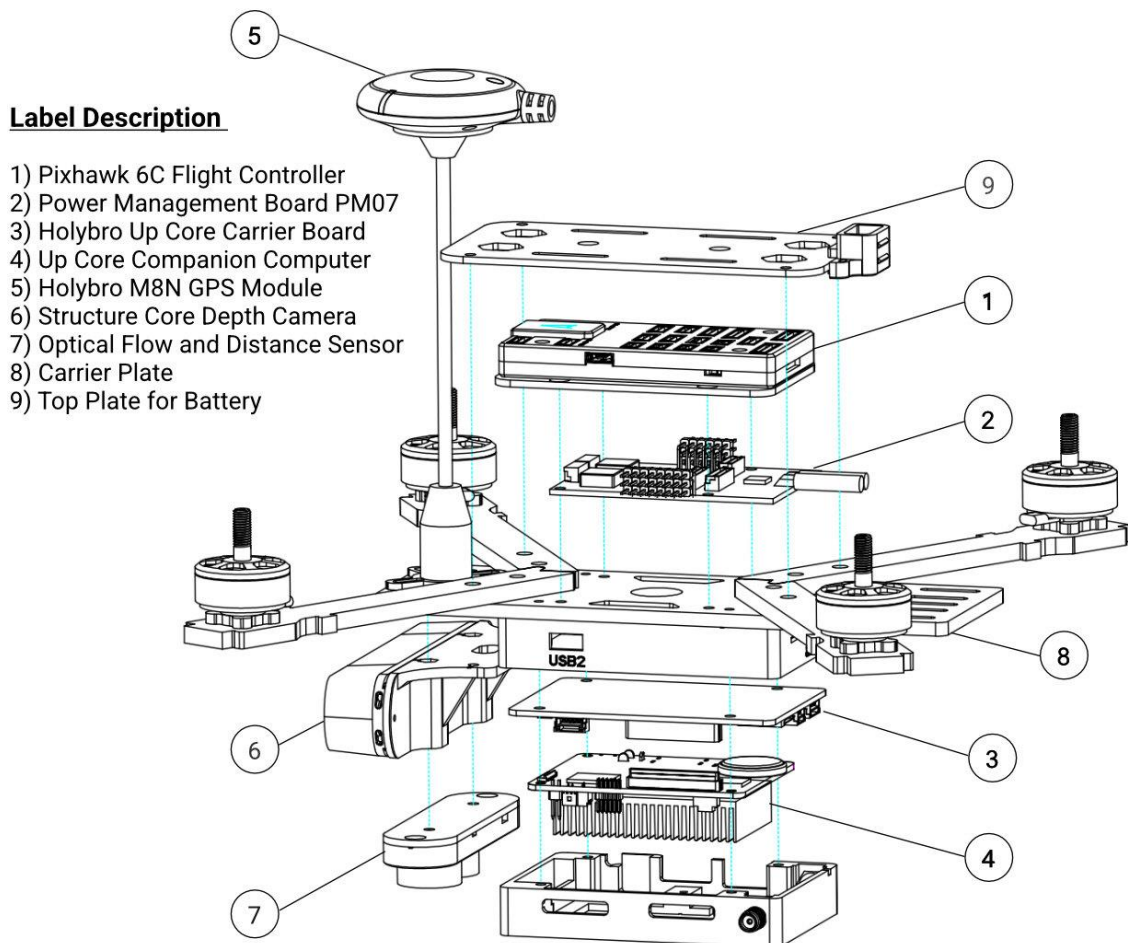
A questo punto, per poter effettuare il primo volo, è necessario avvalersi della chiavetta USB che deve essere inserita nella relativa porta del companion computer. Per poter fare a meno della chiavetta è necessario trasferire il software che essa contiene nella memoria del computer o, in alternativa, creare in autonomia una configurazione software personalizzata. Per capire come ciò sia possibile e quali siano i principali passaggi concettuali necessari, si invita il lettore a fare riferimento al capitolo 2.

Il passaggio successivo è l'installazione e connessione di una batteria carica per alimentare il drone; l'accensione è automatica e quasi immediata. Pochi secondi dopo che il computer dell'autopilota viene acceso questo genera una rete wi-fi a cui il computer con funzione di ground station deve essere collegato. In questo modo, mediante il software QGroundControl, è possibile completare la necessaria procedura di calibrazione di alcuni elementi del drone, quali il ricevitore per il radiocomando appena installato, ma anche la bussola.

L'ultimo passaggio prevede il montaggio delle quattro eliche che devono essere opportunamente fissate adoperando gli appositi dadi forniti.

Il drone è ora pronto per un primo volo di prova.

## 1.2 Hardware e cablaggi



**Figura 4:** Vista esplosa dei component principali del drone

I vari componenti del drone sono fissati ad una struttura realizzata in fibra di carbonio e costituita sostanzialmente da due piastre di supporto. Alla piastra principale sono fissati i bracci di supporto dei rotori e, sul lato inferiore, il companion computer, il sensore optical flow, il sensore di distanza ad infrarossi e la camera Structure Core; sulla faccia superiore è montato il supporto dell'antenna GPS, la power management board ed il flight controller computer, coperti superiormente dalla seconda piastra, che serve per alloggiare ed ancorare la batteria.

## 1.4.1 Flight controller <sup>[5]</sup>

Per consentire ad un qualsiasi veicolo a pilotaggio remoto o autonomo di muoversi ed in generale di espletare le sue funzioni, è necessario che questo sia dotato di un computer principale quale il flight controller per un drone. Il PX4 Vision Autonomy Development Kit è equipaggiato con un computer Pixhawk® 6C che supporta il software dell'autopilota, gestisce alcuni sensori e invia alla scheda di gestione della potenza i comandi destinati ai motori.



Il computer Pixhawk® 6C rappresenta la più recente versione della famiglia di flight controller Pixhawk®, sviluppato in collaborazione con Holybro e PX4. Il Pixhawk® 6C è dotato di un processore STMicroelectronics STM32H743, in combinazione con dei sensori prodotti da Bosch® e InvenSense®. Inoltre, il microcontroller H7 con cui Pixhawk® 6C è equipaggiato contiene un Arm® Cortex®-M7 core capace di raggiungere la frequenza massima di 480 MHz, con 2MB di memoria flash e 1MB di RAM; l'unione di queste caratteristiche conferisce al processore una potenza tale da supportare anche l'esecuzione di algoritmi complessi, sebbene sia comunque necessario affidare al companion computer i processi con elevato peso computazionale.

**Figura 5:** Flight controller Pixhawk 6C

Per quanto riguarda i sensori, Pixhawk 6C è dotato di sensori inerziali (detti IMU) progettati per garantire elevate prestazioni, essere silenziosi ed economicamente efficienti pur garantendo adeguata ridondanza. È inoltre presente un sistema di isolamento dalle vibrazioni per filtrare gli stimoli ad alta frequenza e ridurre quindi il rumore captato con le misurazioni, con un notevole impatto positivo sulle prestazioni di volo nel complesso. Infine, per garantire che i sensori IMU si trovino sempre alla temperatura ottimale di esercizio, è presente un sistema di controllo della temperatura che si avvale di appositi resistori per riscaldare le unità inerziali se necessario.

Si fornisce un sommario delle specifiche tecniche del processore e dei sensori di cui l'unità Pixhawk® 6C è dotata:

- FMU Processor: STM32H743
  - 32 Bit Arm® Cortex®-M7, 480MHz, 2MB memory, 1MB SRAM

- IO Processor: STM32F103
  - 32 Bit Arm® Cortex®-M3, 72MHz, 64KB SRAM
- On-board sensors
  - Accel/Gyro: ICM-42688-P
  - Accel/Gyro: BMI055
  - Mag: IST8310
  - Barometer: MS5611

Perché il flight computer possa svolgere efficacemente il suo compito, è necessario che questo sia adeguatamente connesso con gli altri elementi del drone come spiegato nel paragrafo relativo ai cablaggi. Questo computer è dotato di una porta di alimentazione cui deve essere fornita una corrente elettrica continua con una tensione massima di 6 Volt. L'interfaccia con gli altri elementi del drone dipende dalle svariate porte di cui il computer è dotato:

- 16- PWM servo outputs raggruppati in due porte, 8 nella porta I/O PWM OUT e 8 nella porta FMU PWM OUT
- 3 porte seriali con scopo generico
  - TELEM1 - Full flow control, con corrente massima 1.5A
  - TELEM2 - Full flow control
  - TELEM3
- 2 porte GPS
  - GPS1 - Full GPS port (GPS con safety switch)
  - GPS2 - Basic GPS port
- 1 porta I2C
  - Supporta la calibrazione I2C EEPROM dedicata e collocata sul sensor module
- 2 CAN Buses
  - CAN Bus con controllo silenzioso individuale o ESC RX-MUX control
- 2 porte di debug:
  - FMU Debug
  - I/O Debug
- R/C input dedicata a Spektrum / DSM e S.BUS, CPPM, analog / PWM RSSI
- S.BUS output dedicato
- 2 Power input ports (Analog)

Come a breve spiegato, le porte TELEM verranno in questo caso adoperate per la connessione con il companion computer. Si noti inoltre la presenza di due porte GPS peraltro con un diverso numero di connettori; la ragione è che la



prima porta serve per connettere il cosiddetto Full GPS, il modulo GNSS primario dotato anche di ulteriori funzioni, la seconda porta serve per il modulo GNSS secondario. Per approfondimenti a riguardo si veda la sezione dedicata a seguire.



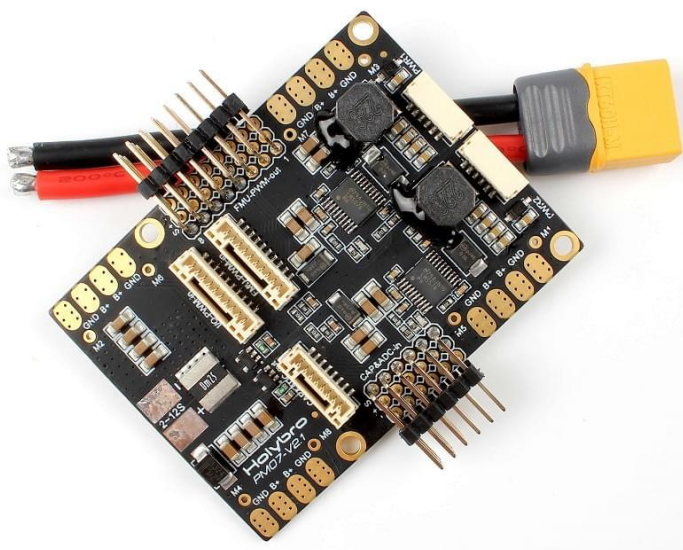
Il computer Pixhawk 6C dispone inoltre di uno slot per schede SD.<sup>[6]</sup> La memoria estraibile, prevista per altro in dotazione, viene utilizzata per registrare i flight logs con i dati di telemetria acquisiti quando il drone viene armato; inoltre, la memoria SD può eventualmente anche essere utilizzata per iniettare nell'autopilota le istruzioni per eseguire missioni precedentemente programmate.

**Figura 6:** Slot per scheda SD

Di default, se nessuna scheda SD è inserita nello slot, l'autopilota invierà un segnale di errore e le funzioni sopra citate non saranno disponibili.

Per quanto riguarda la collocazione dell'autopilota a bordo del drone<sup>[8]</sup>, è importante che questo sia montato in una posizione e con un'orientazione rispetto al veicolo opportune. Nello specifico, la presenza di sensori inerziali all'interno di questo componente fa sì che esso debba essere posizionato per quanto possibile in prossimità del centro di gravità, con la faccia superiore rivolta verso l'alto e la freccia bianca ad indicare la direzione di movimento in avanti. Anche se per il PX4 Vision Autonomy Development Kit il problema non si pone, qualora ci fossero vincoli ad impedire l'installazione dell'autopilota secondo la direzione di default, è possibile concedere una deroga sul requisito a patto che lo stesso autopilota ne sia informato mediante la configurazione di specifiche impostazioni. Infine, per ridurre l'indesiderato effetto delle vibrazioni sui sensori, è possibile ricorrere accorgimenti quali l'interposizione di materiale isolante tra il computer e la struttura a cui questo viene ancorato.

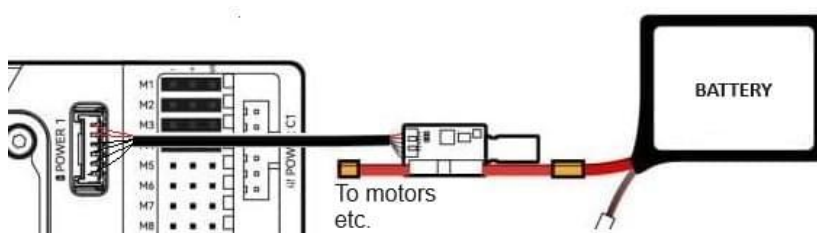
## 1.4.2 Power management board <sup>[8]</sup>



**Figura 7:** Power management board

La scheda di gestione della potenza (detta PM board) di tipo PM07 di cui il PX4 Vision Autonomy Development Kit è dotato serve un duplice scopo: agisce da modulo di potenza e governa la distribuzione della potenza ai vari componenti del drone e di fatto sostituisce due distinti componenti: il power module e la power management board.

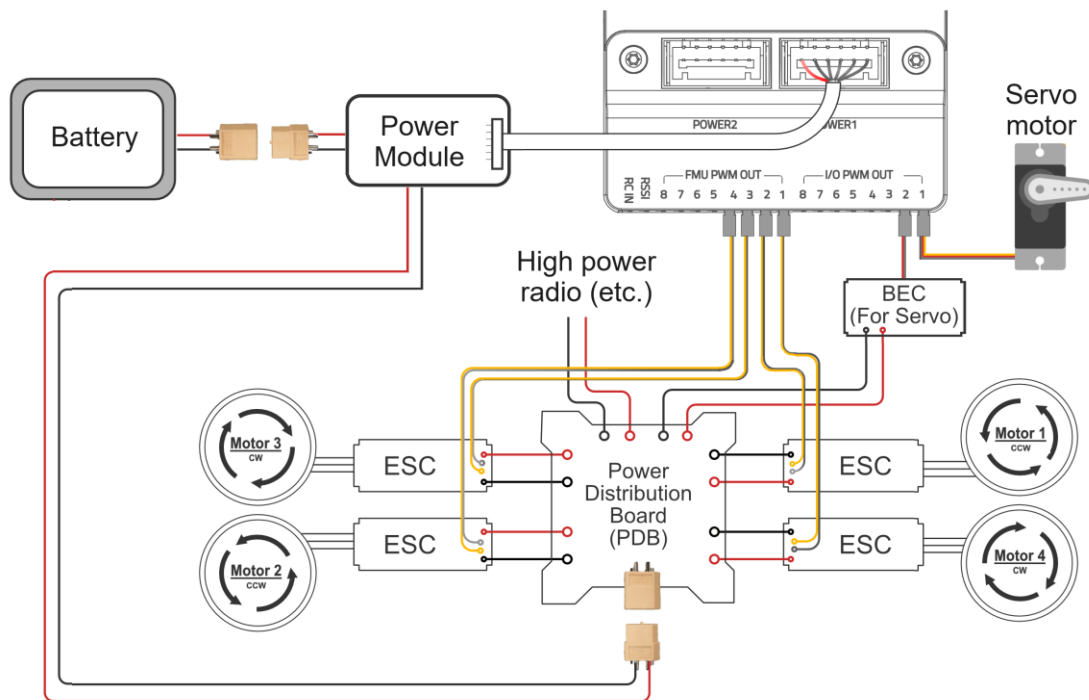
Il power module ha il compito di prelevare l'energia dalla batteria e di suddividerla su due distinti output. Il primo ha lo scopo di alimentare il flight director mediante una corrente che rispetti specifici livelli di tensione e corrente dettati dal computer che, a sua volta, redistribuirà parte della potenza agli altri elementi ad esso collegati, quali sensori e ricevitore GNSS. Il power module trasmette inoltre al flight director informazioni riguardo la tensione della batteria e il flusso complessivo di corrente. Il secondo output di potenza ha invece lo scopo di alimentare tutte le altre utenze di bordo, quali attuatori ed eventuali altri sensori o apparati radio di potenza.



**Figura 8:** Schema esplicativo della funzione del power module

La power management board ha il compito di spartire la potenza ricevuta in input su più output in parallelo. In particolare, questa scheda gestisce l'alimentazione degli attuatori: riceve dei comandi dal flight director che vengono trasformati in livelli di potenza da inviare ai motori e fornisce un riscontro all'autopilota sui flussi di corrente erogati.

Si fornisce quindi uno schema esemplificativo di un circuito come da descrizione, ma è importante ricordare che il power module del PX4 Vision Autonomy Development Kit costituisce un modulo stand alone che accorpa tutte le funzioni in un unico componente.



**Figura 9:** Schema esplicativo della funzione della power distribution board

Infine, la scheda PM07 è dotata di un connettore per la batteria e permette di gestire una corrente elettrica in output fino ad un'intensità massima complessiva di 120 A.



### 1.4.3 Companion computer <sup>[10]</sup>

Il companion computer (o mission computer) è il secondo computer di bordo del drone, è totalmente distinto dal flight controller sebbene con esso sia collegato; tale collegamento può essere implementato via ethernet o via cavo attraverso una porta seriale. Il companion computer ha molteplici scopi: permette di gestire periferiche aggiuntive quali le fotocamere per applicazioni di computer vision, ad esempio, ed al contempo supporta l'esecuzione degli algoritmi computazionalmente pesanti ad esse associati; serve infine a garantire che il controllo del veicolo possa avvenire ad un livello superiore. In primis ciò dipende dal fatto che il companion computer sfrutta un sistema operativo, quale ad esempio Linux, che offre una più ampia compatibilità rispetto al sistema operativo dell'autopilota sia per quanto riguarda la gestione di sensori o payload complessi, ma anche per via dell'ampia disponibilità di software già sviluppato e pronto all'uso. Maggiori dettagli a riguardo, con più specifico riferimento al caso in esame, saranno comunque forniti in seguito.

Dal punto di vista hardware il PX4 Vision Autonomy Development Kit è dotato di un processore Intel UP Core computer compatibile con i principali sistemi operativi. Per le specifiche in dettaglio si veda la sezione dedicata al contenuto del Kit.

### 1.4.4 Modulo GNSS <sup>[11]</sup>

L'utilizzo dei sistemi GNSS è fondamentale per svolgere la maggior parte delle missioni perché permette all'autopilota di determinare la posizione del veicolo nello spazio e, con un certo grado di approssimazione, anche di tracciarne i movimenti. Per questo motivo gli autopiloti PX4 necessitano almeno di un modulo GNSS per funzionare, addirittura il flight controller Pixhawk 6C supporta anche due sensori contemporaneamente. Questi possono essere connessi con l'autopilota sia tramite BUS CAN che UART e, se presenti entrambi, svolgono un ruolo leggermente differente; la presenza del modulo GNSS primario, come si vedrà a breve, è chiamato a svolgere varie funzioni e la sua utilità è indiscutibile, mentre l'installazione del modulo secondario è a discrezione dell'utente dato che questo ha solamente uno scopo di ridondanza.

PX4 supporta un ampio numero di dispositivi GNSS e quasi tutte le principali costellazioni, inclusa SBAS. Siccome l'utilizzo di diversi equipaggiamenti e tecnologie per la determinazione della posizione e l'impiego del GPS aumentato costituiscono gli argomenti principali di questa tesi, si rimanda ad un successivo capitolo un approfondimento su queste tematiche.



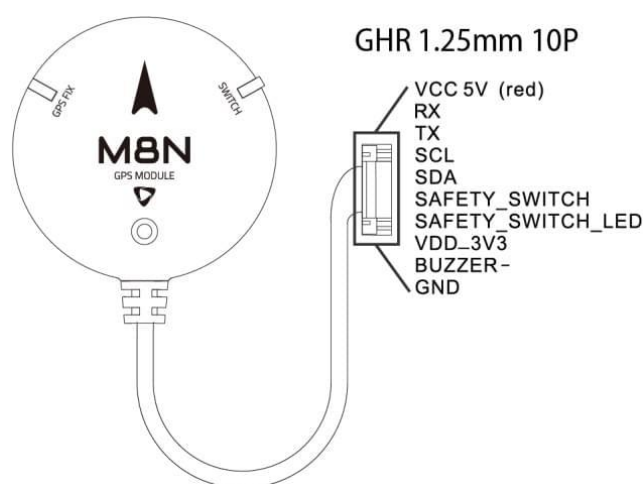
**Figura 10:** Modulo GNSS e connessione con il flight controller

Oltre al ricevitore GNSS, il secondo elemento contenuto nel modulo GNSS è la bussola/magnetometro; questo componente viene utilizzato dall'autopilota per determinare lo stato del veicolo in termini di heading e yaw sfruttando il campo magnetico terrestre come riferimento. La presenza di questo elemento fa sì che il modulo GNSS debba essere montato<sup>[12]</sup> lontano, per quanto possibile, da eventuali fonti di disturbo elettromagnetico quali i motori ed i relativi circuiti di alimentazione, od eventuali altre linee di potenza. Tramite il magnetometro poi, il flight director verifica che l'orientazione spaziale del modulo GNSS sia coerente e cioè che si trovi a faccia in su e anch'esso orientato secondo la direzione di movimento in avanti. Se per qualche motivo è necessario installare questo componente in una posizione ruotata rispetto a quella di default, una configurazione con rotazioni di 45° o multipli verrà automaticamente identificata da PX4, diversamente sarà necessario impostare manualmente la configurazione fornendo i valori relativi agli angoli di Eulero.

Per poter correttamente compensare il moto relativo del sensore rispetto al centro di gravità del veicolo (COG) è necessario fornire all'autopilota la posizione del modulo GNSS rispetto al COG attraverso l'assegnazione di un valore ai tre appositi parametri EKF2\_GPS\_POS\_X, EKF2\_GPS\_POS\_Y ed EKF2\_GPS\_POS\_Z. Questo aspetto acquisisce una crescente importanza al crescere della distanza in questione ed in particolar modo se si ricorre al GPS RTK che raggiunge una precisione centimetrica. La mancata configurazione di questi parametri porterà l'autopilota ad assumere erroneamente che il sensore

GNSS si trovi nel centro di gravità e ad apportare quindi delle correzioni inutili quando al drone dovesse essere richiesto di mantenere la posizione.

I restanti elementi integrati nel modulo GNSS sono il safety switch, il buzzer ed un LED UI RGB<sup>[13]</sup>. Il safety switch è un dispositivo di sicurezza che deve essere inserito prima che il veicolo possa essere armato e volare; per quanto riguarda il buzzer ed il LED, questi hanno il compito di informare l'utente, tramite messaggi acustici e luminosi\*, dello stato del veicolo. Nello specifico, i diversi segnali acustici emessi dal buzzer, affiancati da uno specifico codice di colori assunti dal LED UI, assicurano che l'utente sia immediatamente ed intuitivamente messo a conoscenza del fatto che il veicolo sia acceso, armato, abbia riscontrato un errore o altro. La figura che segue mostra in modo schematico il modulo GNSS Holybro M8N GPS con cui il drone è equipaggiato: sono visibili anche il safety switch ed il LED UI.

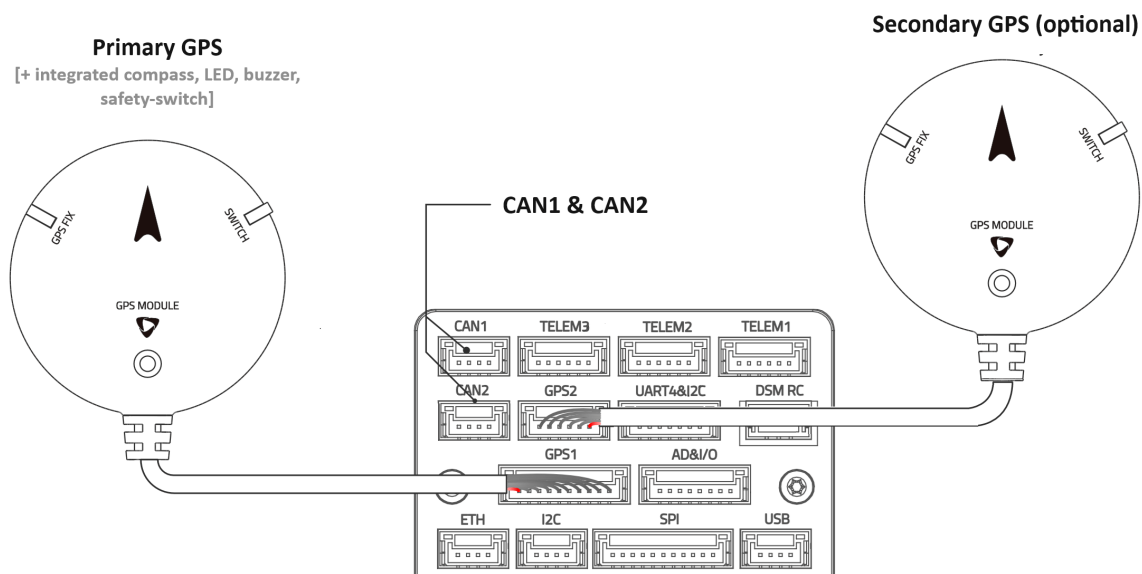


**Figura 11:** Schema del modulo GNSS M8N

Si mostra infine l'importante differenza nel montaggio del modulo GNSS primario dall'eventuale secondario. Nello specifico, come appena spiegato, il modulo primario supporta un maggiore numero di funzioni e deve quindi essere connesso all'apposita porta dell'autopilota con un maggiore numero di terminali; al contrario il modulo secondario viene utilizzato puramente come ricevitore GNSS di scorta e deve quindi essere collegato alla porta secondaria che, con un ridotto numero di terminali, non supporta le altre funzioni. Per la massima chiarezza si fornisce una figura rappresentativa di questa configurazione.

---

\*Nota: sono omesse a tal proposito le informazioni di dettaglio. Per approfondimenti si invita a fare riferimento alla documentazione ufficiale. <sup>[14]</sup>



**Figura 12:** Collegamento dei moduli GNSS primario e secondario con il flight controller

### 1.4.5 Camera [15]



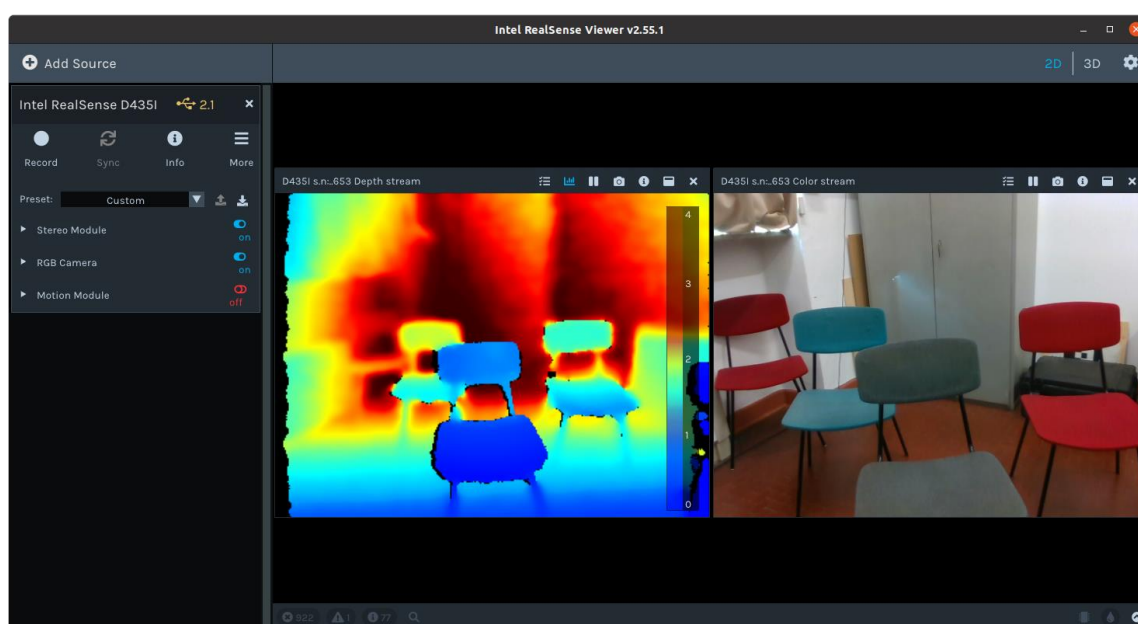
**Figura 13:** Camera Occipital Structure Core

Per l'implementazione di funzioni di computer vision è necessario l'utilizzo di una camera dedicata con alcune caratteristiche specifiche, quali la presenza di più fotocamere per consentire la ricostruzione dell'ambiente osservato anche nella terza dimensione. Il PX4 Vision Autonomy Development Kit è infatti dotato di una camera di profondità Occipital Structure Core montata frontalmente e connessa al companion computer responsabile della sua gestione e dell'implementazione delle funzioni da essa dipendenti.

Questo sensore è in pratica costituito da tre fotocamere, un'unità inerziale ed un processore. La camera centrale opera nel campo del visibile ed ha un campo visivo di 160°, essa può essere utilizzata ad esempio per applicazioni

quali l'individuazione di target segnalati con marker appositi. Ci sono poi le due camere più esterne che operano invece su frequenze più basse, nel campo dell'infrarosso; le loro immagini combinate permettono, grazie all'elaborazione da parte del processore, di ottenere una visione tridimensionale che consente quindi di determinare la distanza a cui si trovano gli ostacoli inquadrati: è proprio per questo motivo che è necessario che le camere siano due e cioè per poter realizzare la visione stereoscopica, basandosi sullo stesso principio tipico anche dell'apparato visivo umano. Nello specifico, il componente con cui il drone è equipaggiato è una depth camera, questo significa che grazie ai sensori ed al processore di cui è dotata è capace di generare in tempo reale una mappa di profondità che viene descritta, a livello informatico, attraverso una point cloud. Il ruolo dell'unità inerziale, di cui la camera è dotata, è poi quello di stabilizzare l'immagine per poter raffinare la percezione della terza dimensione, soprattutto nel caso in cui la camera si muova. Di fatto, la presenza di sensori inerziali all'interno della camera permette di garantire un migliore allineamento della point cloud, molto utile nelle applicazioni di tracking o SLAM (acronimo per simultaneous localization and mapping).

Visualizzando a schermo (figura 14) le immagini della camera è possibile comprendere appieno le potenzialità e l'importanza di questo sensore:



**Figura 14:** Test vista RGB e di profondità della camera

All'immagine ottenuta nel campo del visibile viene affiancata l'immagine ad infrarossi in cui, grazie alla visione stereoscopica, è possibile visualizzare la profondità resa attraverso la scala cromatica; i colori freddi indicano oggetti più vicini, mentre quelli più caldi sono utilizzati per gli oggetti più lontani. La

camera ha un range di funzionamento: le superfici o gli oggetti troppo vicini o troppo lontani appaiono a schermo come macchie nere, ma non è questo il caso nell'immagine, per la quale questa situazione è stata volutamente evitata.

#### 1.4.6 Optical flow e distance sensor <sup>[16]</sup>

Lo scopo dell'optical flow è quello di fornire un metodo sempre affidabile per la determinazione della velocità che sia indipendente dalla possibilità di connettersi ad una rete GNSS; il beneficio ottenuto dall'adozione di questa tecnologia è quindi particolarmente evidente in caso di operazioni sotterranee, all'interno di edifici e più in generale in ogni scenario operativo che per qualsiasi motivo veda negata la possibilità di sfruttare i sistemi di navigazione satellitari.

Dal punto di vista hardware, l'optical flow necessita dell'impiego di due sensori orientati verso il basso: una camera ed un sensore che misuri la distanza, di tipo range finder o preferibilmente LiDAR (il secondo è tipicamente più accurato ed affidabile). In alcuni casi è anche possibile ricorrere ad un unico dispositivo che integri entrambi i dispositivi, ma così non è per il PX4 Vision Autonomy Development Kit. A bordo di questo drone è installato come un unico blocco un elemento che combina al suo interno i due distinti componenti: il sensore optical flow ed il distance sensor.



**Figura 15:** Sensore Optical flow



**Figura 16:** Distance sensor

Il modulo PMW3901<sup>[17]</sup>, nell'immagine di sinistra, integra un sensore di tracking ed una piccola unità di calcolo. Il sensore di tracking è molto simile a quello con cui sono equipaggiati i mouse, ma adattato per lavorare ad una distanza a partire da 80 mm fino all'infinito; l'unità di calcolo invece ha lo scopo di determinare gli spostamenti valutando il movimento dei pixel in frame successivi. Come in parzialmente già anticipato, questo componente necessita di essere accoppiato con un range finder quale il Lanbao PSK-CM8JL65-CC5

ToF Infrared Distance Measuring Sensor<sup>[18]</sup> mostrato nell'immagine di destra; quest'ultimo ha un range di funzionamento tra 0,17 m e 8 m con una risoluzione millimetrica.

Durante il suo funzionamento, il sensore genera dei dati in output che descrivono il movimento rispetto al suolo secondo i due assi principali del piano orizzontale. Questi dati vengono sfruttati dall'autopilota a valle della combinazione con altre informazioni di velocità provenienti da altre fonti a bordo. Il processo è gestito dall'estimator che si avvale di un filtro di Kalman esteso, identificato per sintesi dalla sigla EKF2.

Per quanto riguarda il montaggio del sensore a bordo del drone, è fondamentale che questo sia rivolto verso il basso, ma per il resto può essere posizionato con relativa libertà. La posizione ottimale sarebbe in prossimità del centro del veicolo, ma una collocazione alternativa è possibile purché l'utente modifichi coerentemente l'impostazione dei parametri di offset necessari al filtro di Kalman esteso; in particolare i parametri EKF2\_OF\_POS\_X / Y / Z rappresentano la posizione del sensore a bordo del veicolo secondo gli assi corpo. Una riflessione analoga riguarda l'orientazione del sensore in termini di angolo di yaw: sul sensore è presente una piccola tacchetta che deve essere rivolta nella stessa direzione del retro del veicolo, qualora ciò non fosse possibile, il montaggio del componente in posizione ruotata è permesso a patto che sia di conseguenza modificato dall'utente il valore della variabile dell'autopilota SENS\_FLOW\_ROT.

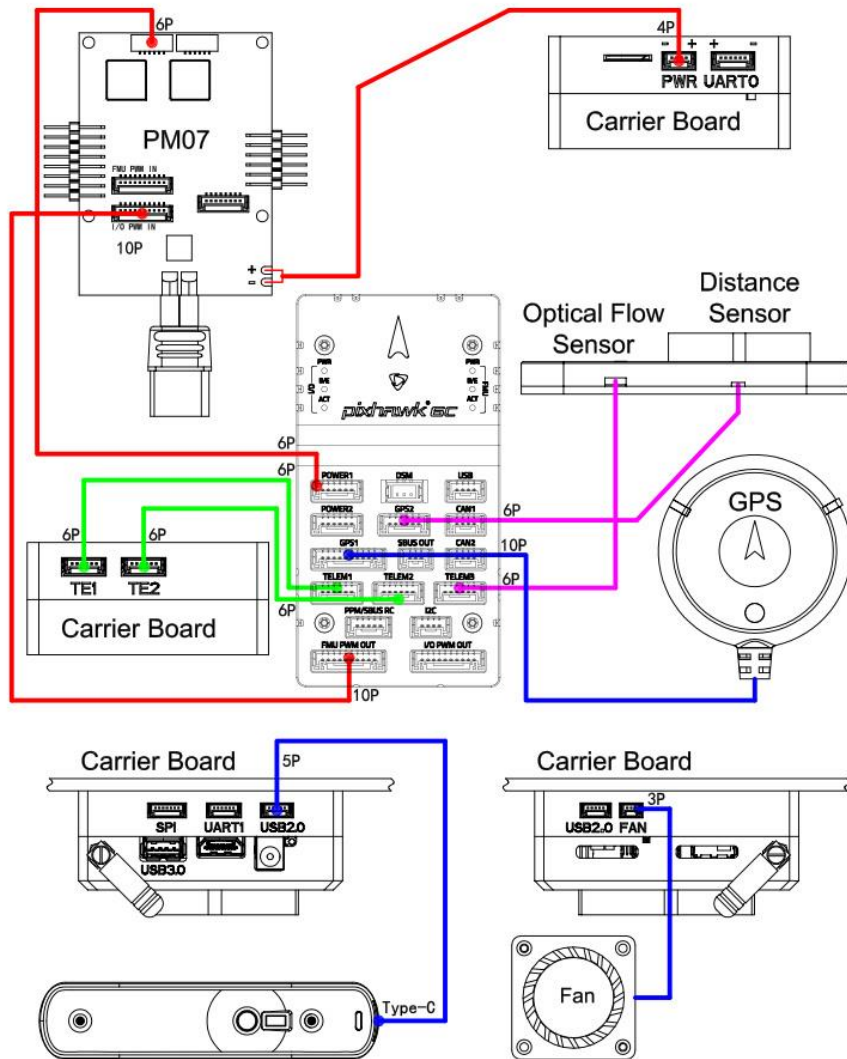
#### 1.4.7 **Cablaggi** <sup>[19]</sup>

Per garantire il corretto funzionamento di tutti i componenti del kit, sia per quanto riguarda l'alimentazione elettrica che le interfacce per lo scambio di informazioni, dati o parametri, è fondamentale assicurarsi che sussistano tutti i necessari collegamenti tra i diversi elementi hardware.

In quest'ottica non si può prescindere da una descrizione dei cablaggi del drone proprio per via del fondamentale ruolo che le connessioni tra gli elementi costitutivi rivestono all'interno di qualsiasi sistema.

Lo schema fornito ha quindi il compito di mostrare sinteticamente in che modo devono essere tra loro collegati tutti i componenti ed equipaggiamenti di bordo che sono stati testé descritti.





**Figura 17:** Schema dei cablaggi

A rivestire un ruolo di primaria importanza sono i collegamenti, rappresentati in rosso nello schema dei cablaggi, della scheda di gestione della potenza elettrica con i due computer di bordo: il flight controller (Pixhawk 6c al centro della figura) e la carrier board del companion computer. Quando si intende far volare il drone, la batteria viene connessa alla scheda di potenza che ha il compito di alimentare i motori, i due computer e le altre utenze ad essi collegate: sono infatti i computer stessi ad agire da ulteriori distributori della potenza presso i sensori ed i componenti a bassa potenza ad essi collegati. Quando si necessita di operare con uno o entrambi i computer a scopo di programmazione e/o configurazione, è possibile garantire l'accensione di entrambi i computer indipendentemente mediante l'utilizzo di appositi alimentatori esterni, senza ricorrere all'utilizzo della batteria e di fatto bypassando la scheda di gestione della potenza. Osservando le connessioni tra il flight controller e la scheda di potenza, è possibile notare come sia presente



un doppio cablaggio, questo perché, attraverso la porta power1 il computer pixhawk riceve la corrente elettrica necessaria al suo funzionamento, mentre attraverso la porta FMIJ PWM OUT l'autopilota invia alla scheda di potenza i comandi relativi all'alimentazione dei motori, che vengono da questa ricevuti attraverso la porta I/O PWM IN.

Le linee verdi mostrano le connessioni, attraverso le porte telem1 e telem2, del flight controller con il companion computer utili allo scambio di dati ed in particolar modo all'invio di istruzioni all'autopilota da parte del companion computer. Si tratta di un collegamento non strettamente indispensabile al volo di per sé, ma invece fondamentale per consentire l'implementazione di quelle funzioni che necessitano del companion computer per essere implementate oltre che per l'interfaccia con i componenti collegati a questo computer e non direttamente al flight controller. È comunque necessario precisare che, qualora l'utente lo desiderasse, è anche possibile sostituire la connessione via cavo con un ponte wireless attraverso alcune operazioni di configurazione; nel caso in esame si è preferito sfruttare la certa robustezza del collegamento via cavo ma questa scelta è in ogni caso arbitraria.

Il companion computer è poi collegato con la camera Structure Core tramite porte USB per permettere contestualmente l'alimentazione della camera ma anche lo scambio di informazioni con essa; il collegamento con la ventola di raffreddamento ha invece il solo scopo di fornire la potenza elettrica necessaria al funzionamento di quest'ultimo componente.

Per quanto riguarda il flight controller computer, questo è connesso con l'antenna GPS da cui ottiene informazioni di posizione, ma anche ai sensori optical flow e la camera di distanza ad infrarossi utilizzati per migliorare le prestazioni in volo come già spiegato; la determinazione della posizione e lo sfruttamento dei sensori a tale proposito sarà oggetto di approfondite argomentazioni future. Si noti comunque che il distance sensor viene connesso attraverso la porta dedicata al GPS secondario, ragion per cui non sarà eventualmente possibile installare tale componente se non mediante il BUS CAN.

## 2) Autopilota e framework setup

Perché un drone possa funzionare correttamente, non è certo sufficiente assemblare e connettere i suoi componenti hardware, ma è necessario che il sistema sia completato integrando anche gli elementi software. Tra questi, il più importante è certamente l'autopilota, responsabile della gestione di tutte le funzionalità più immediatamente connesse al volo: l'acquisizione dei dati di volo raccolti dai sensori e la loro rielaborazione per ottenere un output per i motori.

Naturalmente, la complessità del software di bordo andrà crescendo a mano a mano che si richiede al veicolo di espletare un maggior numero di funzioni; proprio per questo motivo il PX4 Vision Autonomy Development Kit è equipaggiato con un mission computer. Lo scopo di questo componente è infatti quello di permettere la creazione di un framework personalizzato o personalizzabile da parte dell'utente, che possa servire all'implementazioni di ulteriori funzioni.

### 2.1 Autopilota <sup>[20][21][22]</sup>

Il drone il PX4 Vision Autonomy Development Kit è equipaggiato con un autopilota PX4; PX4 è una piattaforma software open-source utilizzata per il controllo di veicoli autonomi, come droni, rover, robot sottomarini e velivoli senza pilota. È sviluppata principalmente per fornire un autopilota completo e flessibile, permettendo a questi veicoli di volare o muoversi autonomamente, seguendo comandi impartiti da un radiocomando o missioni predefinite.

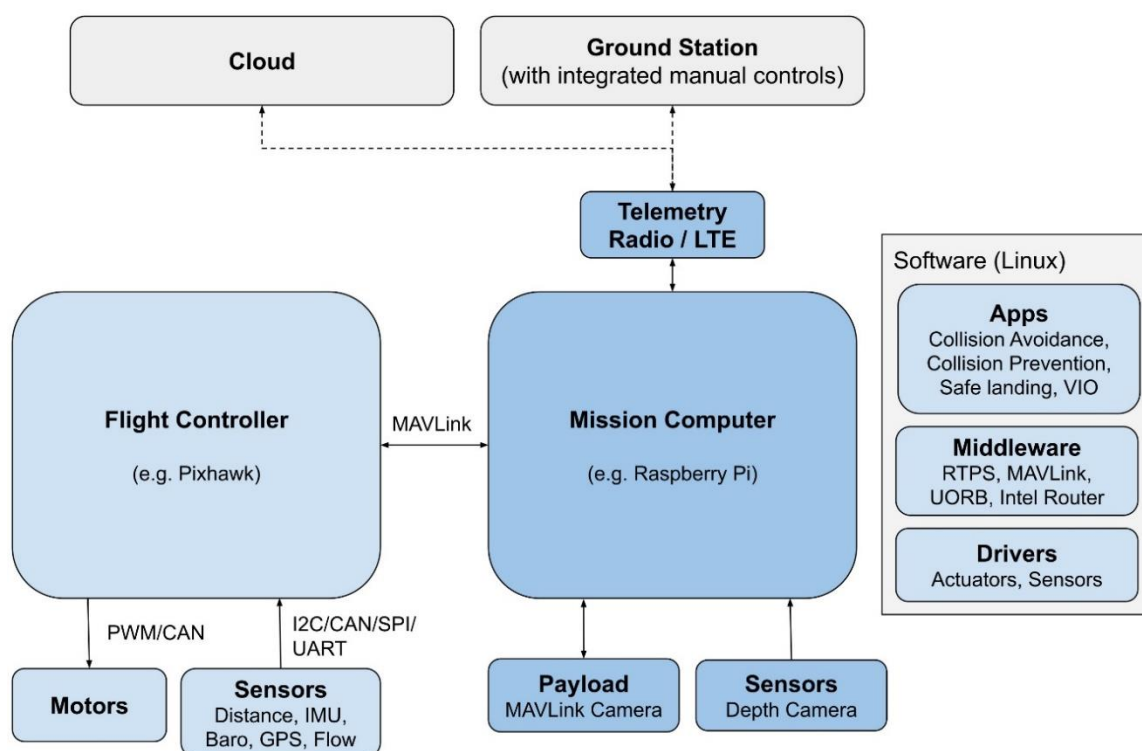
Per quanto riguarda l'architettura di PX4, è bene precisare che ne esistono di due tipologie: la più semplice si basa sull'utilizzo di un solo computer chiamato flight controller, che si interfaccia quindi con tutti gli altri elementi del veicolo. Il PX4 Vision Autonomy Development Kit presenta invece, come già anticipato, un'architettura con due computer; il secondo, detto companion computer o anche mission computer, ha il compito di gestire l'interfaccia con il payload, alcuni sensori e, attraverso una connessione radio, con la ground station.

Nello specifico, il flight controller esegue il software dell'autopilota PX4, mentre il companion computer ha il compito di supportare le funzioni più avanzate come, ad esempio, quelle di computer vision. I due computer sono collegati attraverso una veloce connessione seriale e comunicano tipicamente

attraverso il protocollo MAVLink o  $\mu$ RCE-DDS, i due protocolli e le loro differenze verranno approfonditi in una sezione dedicata a seguire. Anche le comunicazioni con la ground station ed eventualmente con il cloud sono solitamente gestite dal companion computer.

Un grande vantaggio dell'architettura con due computer è che le piattaforme PX4 adoperano solitamente sistemi operativi Linux sul companion computer. A differenza del Sistema operativo NuttX impiegato dal flight controller, Linux risulta migliore e più flessibile per lo sviluppo software in generale; è anche importante notare che uno sviluppatore che si appoggia a questo sistema operativo può contare sul supporto di una vasta community, oltre al fatto che è possibile reperire in rete software già scritto utile per integrare funzioni di computer vision, comunicazione, connessione al cloud e driver per hardware.

È fornito uno schema di questa seconda architettura in figura 18.



**Figura 18:** Architettura dell'autopilota con companion computer

Lo schema è fornito a solo titolo di esempio e si riferisce ad un'architettura generica, con lo scopo di mostrarne gli elementi principali e le relative connessioni. Si noterà infatti che, come già in precedenza specificato, il PX4 Vision Autonomy Development Kit utilizza un processore Intel UP Core come mission computer e non un Raspberry Pi: per informazioni accurate riguardo ai

componenti hardware si invita a fare riferimento alla sezione dedicata. Anche per quanto riguarda le applicazioni, il middleware o i protocolli di comunicazione informazioni più specifiche e precise verranno fornite in seguito.

Si noti anche che la possibilità di connessione remota attraverso LTE, sebbene rappresentata in figura, non è prevista di default; si tratta esclusivamente di una potenzialità che lo sviluppatore può sfruttare a seguito di alcune necessarie operazioni di implementazione e configurazione.

### 2.1.1 Software

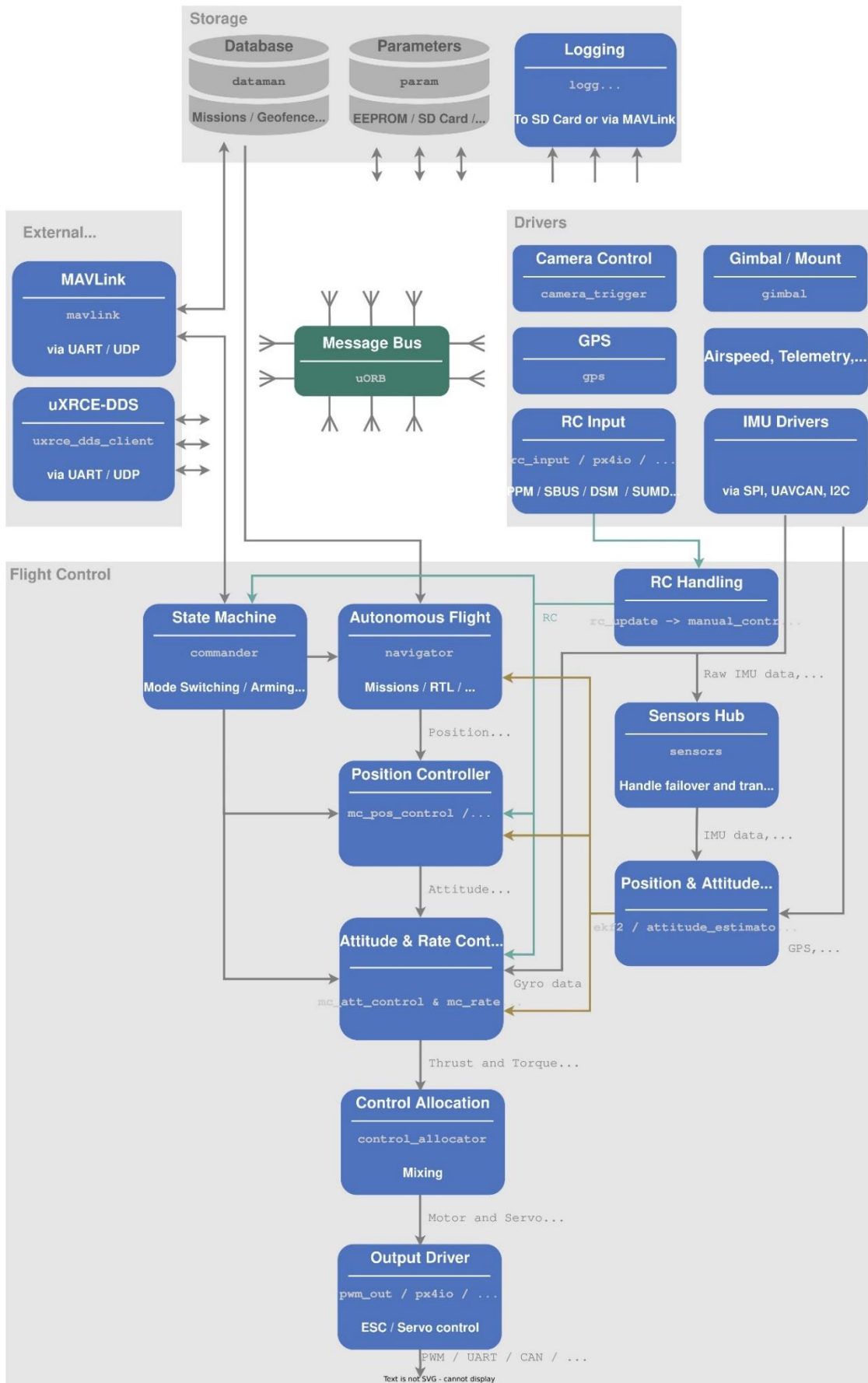
Dal punto di vista software, l'autopilota PX4 è costituito da due elementi detti layer: il flight stack rappresenta il vero e proprio flight control system mentre il middleware è un generico layer robotico, che può cioè essere impiegato da qualsiasi robot, per consentire comunicazioni sia interne che esterne e permettere l'integrazione dell'hardware.

Tutti i velivoli basati su PX4 condividono con altri tipi di robot e veicoli a guida autonoma il medesimo codice alla base. Il design dell'intero sistema è reattivo, più precisamente si intende dire che possiede tre caratteristiche in particolare: tutte le funzionalità sono suddivise ed espletate da componenti riutilizzabili ed intercambiabili, la comunicazione avviene grazie al passaggio di messaggi secondo una modalità asincrona e che il sistema è in grado di adattarsi a diverse condizioni di carico di lavoro.

Per meglio comprendere l'architettura di alto livello del software di PX4, è fornita una figura con uno schema a blocchi dettagliato (figura 19 a pagina 37). Nella parte superiore dello schema sono mostrati i componenti del middleware, mentre la parte bassa è dedicata al flight stack.

Come si può dedurre dallo schema l'architettura è modulare, infatti il codice sorgente è suddiviso in moduli o programmi, ognuno a sé stante, mostrati in figura con un blocco ciascuno. Grazie a questa caratteristica, ciascun blocco può essere sostituito celermente e con facilità in qualsiasi momento.

Le frecce indicano poi il flusso delle informazioni rappresentativo delle connessioni principali tra i moduli; nella realtà pratica, infatti, il numero di connessioni è significativamente maggiore, tanto più perché l'accesso ad alcuni dati e parametri è necessario al funzionamento della maggior parte dei moduli.



**Figura 19:** Schema dell'architettura di alto livello del software di PX4

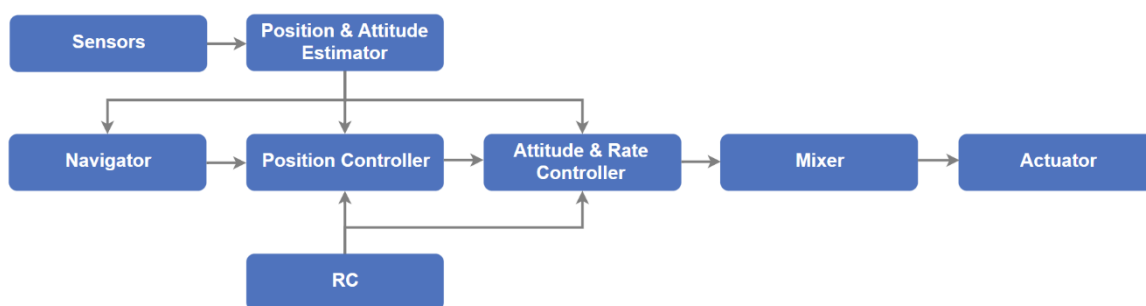
La comunicazione tra i moduli è garantita da un bus per i messaggi chiamato uORB (al centro della figura). Questo sistema funziona grazie all'impiego, da parte di ciascun modulo, di una tecnica di comunicazione di tipo publish-subscribe, con alcuni importanti ed evidenti vantaggi: il sistema consente un'elevata reattività, è asincrono ed i dati sono immediatamente aggiornati appena disponibili, tutte le operazioni e comunicazioni sono parallelizzate e ciascun componente può avere un accesso ai dati sicuro da qualsiasi tipo di minaccia.

Queste caratteristiche non sono casuali, tanto che rappresentano alcuni dei principali punti di forza di PX4 e la ragione del suo successo e ampia diffusione commerciale.

## 2.1.2 Flight stack

Il flight stack è il cuore dell'autopilota e contiene gli algoritmi necessari alla guida, navigazione e controllo di veicoli a guida autonoma quali i droni. Include controllori adatti a velivoli ad ala fissa, multicotteri e VTOL, ma anche degli algoritmi di stima di assetto e posizione.

Il diagramma che segue mostra una panoramica dei principali blocchi costitutivi del flight stack. Esso descrive la pipeline completa a partire da sensori, radiocomando e sistema di controllo del volo autonomo (navigator), fino agli attuatori, siano essi motori o servo controlli.



**Figura 20:** Schema a blocchi del flight stack

Il blocco estimator riceve in input i dati da uno o più sensori, li combina e li utilizza per determinare lo stato del veicolo. Per processare i dati, questo modulo si avvale di algoritmi complessi quale il filtro di Kalman esteso (EKF).

I blocchi controller agiscono sulle variabili di controllo con l'obiettivo di modificarne il valore fino a raggiungere il setpoint ricevuto in input. Il setpoint può provenire dal radiocomando o dal blocco di navigazione e deve essere confrontato con le variabili di stato misurate o calcolate a bordo attraverso una stima. L'output del controllore è la correzione necessaria a raggiungere lo stato desiderato.

Il blocco mixer è così nominato con un chiaro riferimento alla tradizione elicotteristica, il suo ruolo è infatti quello di interpretare i comandi ricevuti in termini di ratei o assetti e di tradurli in istruzioni sui livelli di potenza da inviare ai singoli motori. Al tempo stesso il mixer ha anche il compito di assicurarsi che i livelli di potenza richiesti siano contenuti all'interno di un inviluppo i cui limiti non devono essere superati per non minare alla stabilità del veicolo. In ogni caso è bene specificare che la traduzione operata dal mixer dipende dal tipo di veicolo ed in particolare da aspetti quali la posizione dei motori rispetto al centro di massa od il momento di inerzia del veicolo stesso.

### **2.1.3 Middleware**

Il middleware funge da ponte tra l'hardware e i moduli software di alto livello, in pratica ha lo scopo di gestire l'interfaccia tra il flight stack e gli elementi necessari al suo funzionamento. Infatti, i principali elementi costitutivi del middleware sono i driver necessari alla gestione dei sensori incorporati (quali GPS, giroscopi, eccetera) insieme alle interfacce necessarie alla comunicazione con il mondo esterno; con ciò si intende la connessione con il companion computer, con la ground station ed il bus messaggi uORB attraverso il paradigma publish-subscribe.

Inoltre, il middleware include un layer di simulazione che consente l'esecuzione di PX4 all'interno del desktop di un normale sistema operativo, ciò per permettere il controllo di un modellino computerizzato di un veicolo all'interno di un ambiente di simulazione.

### **2.1.4 Sistema operativo**

PX4 può essere eseguito sfruttando vari sistemi operativi purché aderenti allo standard di compatibilità POSIX, nello specifico il requisito riguarda l'interazione delle applicazioni con il sistema operativo (POSIX-API). Esempi di sistemi operativi adeguati sono Linux, macOS, NuttX o QuRT, ma è necessario il soddisfacimento di un'altra condizione: è necessario che il sistema operativo



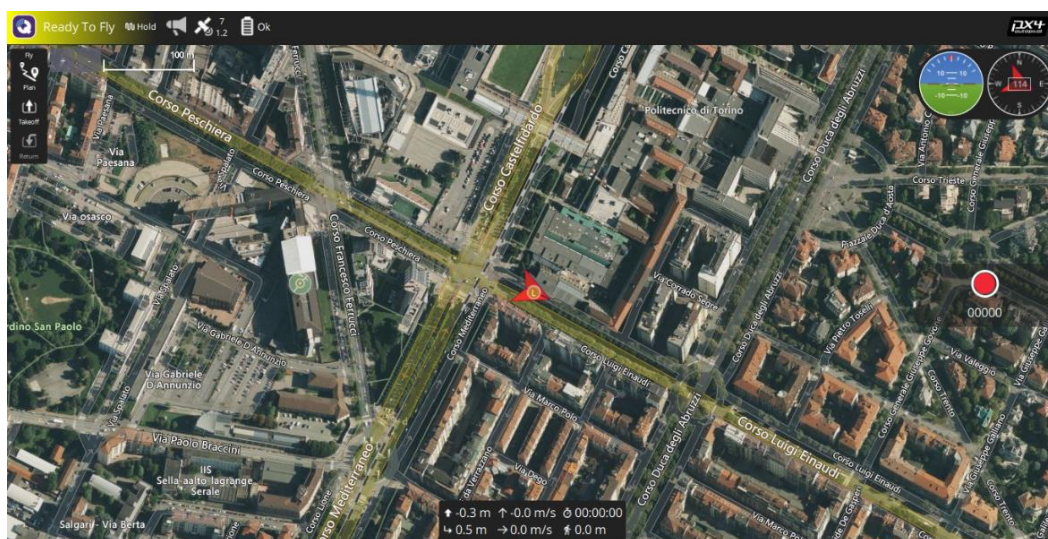
abbia uno scheduling real-time, come ad esempio il comune First-In, First-Out (FIFO).

Per l'esecuzione di PX4 su una flight-control board (come pixhawk ad esempio), il sistema operativo più utilizzato è NuttX. Si tratta di un sistema operativo real-time, open source, molto leggero, efficiente e molto stabile, specificatamente sviluppato per garantire la gestione precisa e sicura delle risorse hardware, come sensori e attuatori; tutte caratteristiche fondamentali per l'autopilota di un multicottero unmanned.

## 2.2 Interfaccia e gestione dell'autopilota mediante ground station

Per consentire all'utente di interfacciarsi con il veicolo ed il suo autopilota, è già stata esposta l'esigenza di impiegare un computer desktop con funzione di ground station; su questo deve essere installato il software QGroundControl<sup>[23]</sup>, uno strumento di interfaccia intuitivo e specifico per lo scopo.

QGroundControl comunica con il drone attraverso un ponte radio bidirezionale, che consente di avere un accesso in tempo reale ai dati di volo, oltre che di controllare e configurare il veicolo ed i suoi componenti. Se l'hardware della ground station lo consente è addirittura possibile pilotare il veicolo. Infine QGroundControl rappresenta uno strumento utile alla pianificazione, esecuzione e monitoraggio di missioni autonome, infatti il software mostra, nella schermata principale, una mappa con marcata la posizione del drone in tempo reale.

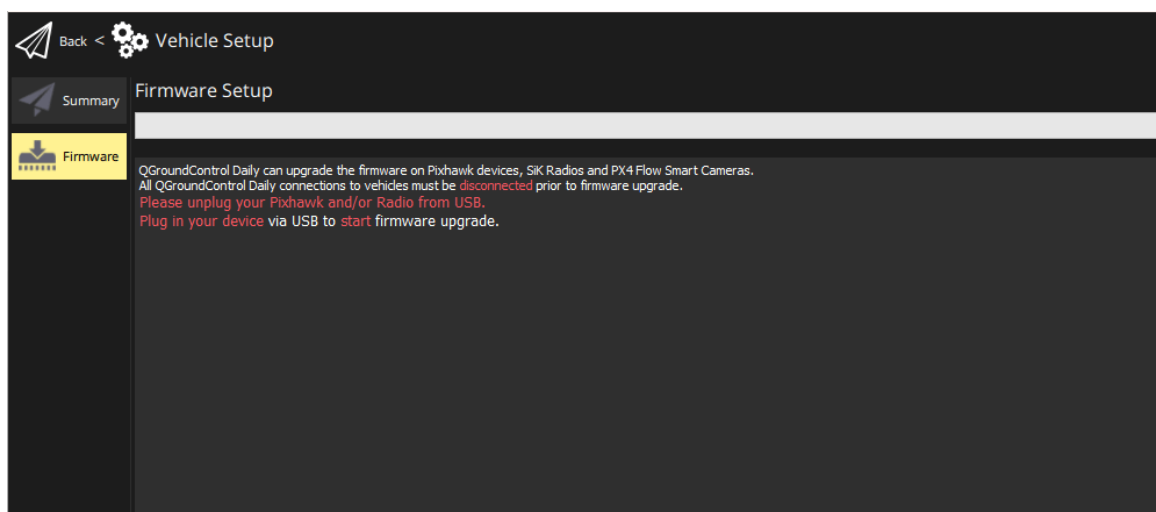


**Figura 21:** Schermata principale di QGroundControl



Dalla schermata principale è possibile accedere ad un grande numero di finestre con scopi specifici, per brevità si invita il lettore a consultare la guida ufficiale per informazioni approfondite; in questa sede verranno elencati esclusivamente gli aspetti più rilevanti. In primis, come già anticipato nella sezione dedicata al setup per il primo volo, attraverso QGroundControl è necessario eseguire la calibrazione di alcuni sensori quali la bussola/magnetometro. Più in generale, il software è compatibile con una vasta gamma di veicoli a pilotaggio remoto, ragion per cui è fondamentale accertarsi che sia impostato per interagire con il tipo di veicolo corretto. Oltre a questo, esiste una schermata dedicata alla visualizzazione ed eventuale modifica dei parametri dell'autopilota PX4; ogni volta che si menziona l'esigenza di verificare o configurare il valore di alcuni parametri è tramite QGroundControl che l'utente deve interfacciarsi con il veicolo. Questo software è poi fondamentale per avere un riscontro sullo stato del drone prima, durante e dopo il volo: è possibile verificare che il veicolo sia pronto e possa essere armato o visualizzare eventuali messaggi di errore, durante il volo le informazioni di posizione e telemetria sono ricevute e visualizzate in tempo reale ed infine, dopo il volo, è possibile accedere ai log file per condurre analisi dettagliate a partire dai dati raccolti in volo. Infine, è tramite QGroundControl che è possibile accedere alla console dell'autopilota per eseguire operazioni da linea di comando, apportare modifiche o anche semplicemente per installare aggiornamenti del firmware.

Per quanto riguarda quest'ultimo aspetto, l'aggiornamento dell'autopilota<sup>[24]</sup> è possibile attraverso una semplice procedura guidata: è sufficiente aprire la pagina di QGroundControl dedicata e, verificata la presenza di aggiornamenti disponibili, eseguire le istruzioni per effettuare l'aggiornamento vero e proprio.



**Figura 22:** Schermata di aggiornamento del firmware in QGroundControl

Le criticità legate all'installazione di una diversa versione del firmware di PX4 non riguardano quindi la procedura di aggiornamento di per sé, ma la verifica ed implementazione delle eventuali modifiche che ciò rende necessarie dal punto di vista di altri elementi del drone. Per esempio, alcune applicazioni implementate sul companion computer possono richiedere lo scambio di messaggi con PX4 ed è quindi necessario verificare che, con una diversa versione dell'autopilota, l'architettura dei suddetti messaggi sia invariata oppure, in caso contrario, apportare le dovute modifiche.

La connessione tra drone e ground station può avvenire in due modi, con il link primario o con il link secondario. Il link primario si basa sulla trasmissione via radio ed è l'unico utilizzabile durante il volo. Quando invece il veicolo è a terra e magari persino spento, il link secondario permette l'interazione di QGroundControl con l'autopilota sfruttando una connessione via cavo USB. Naturalmente in questo secondo caso alcune funzioni saranno limitate, ad esempio non sarà possibile impartire comandi di manetta né far volare il veicolo in nessun modo.

Infine, è importante sottolineare che QgroundControl può supportare anche più di un veicolo simultaneamente consentendo quindi di gestire voli in formazione.

Per quanto riguarda il pilotaggio vero e proprio, come già detto è possibile impartire comandi al drone dalla ground station se questa lo consente, ma non si tratta in generale della soluzione più pratica o più diffusa: nella maggior parte dei casi il volo manuale viene gestito ricorrendo all'utilizzo di un apposito radiocomando la cui configurazione è peraltro possibile mediante QGroundControl.

## 2.3 Framework setup

Come già anticipato, il drone PX4 Vision Autonomy Development Kit è equipaggiato con un companion computer inizialmente privo di qualsiasi elemento software, a cominciare dallo stesso sistema operativo. Per poter effettuare i primi test di volo è possibile adoperare il software contenuto nel driver USB fornito, ma questa soluzione non garantisce il raggiungimento di livelli prestazionali particolarmente spinti. Anche la guida online suggerisce infatti di trasferire il contenuto della chiavetta USB nella memoria del computer: la significativamente maggiore velocità di accesso alla memoria interna rispetto al dispositivo esterno garantisce infatti un'esecuzione più rapida e fluida del software.

In pratica però non è questa la soluzione che è stata adottata nel caso in oggetto. Al fine di ottenere un grado di personalizzazione delle funzioni del drone maggiore e soprattutto per avvicinarsi maggiormente allo stato dell'arte, si è preferito utilizzare software più recenti o versioni più aggiornate sia per l'autopilota che per il sistema operativo del companion computer. L'obiettivo è quindi quello di ottenere un drone ottimamente configurato, che possa agilmente offrirsi a ulteriori modifiche e che sia compatibile con i framework più recenti quale, ad esempio, ROS 2. È proprio a tal proposito che si è scelto di installare il sistema operativo Linux Ubuntu 20.04 LTS invece della versione 18.04 fornita sulla chiavetta, ma l'argomento verrà a breve chiarito e approfondito.

Prima di passare alla descrizione delle fasi di configurazione, dei singoli elementi del framework e la loro interfaccia, è opportuno richiamare quale sia il contenuto del driver USB fornito con il kit per poter meglio comprendere i passaggi necessari per replicare manualmente una configurazione personalizzata, che possa in ogni caso garantire tutte le funzioni inizialmente previste. Il contenuto del supporto USB fornito è quindi il seguente:

- Sistema operativo Linux Ubuntu 18.04 LTS
- ROS Melodic Morenia
- Occipital Structure Core ROS driver
- MAVROS
- PX4 Avoidance

L'elenco descrive quindi gli elementi che dovranno essere installati sul companion computer e funge perciò da riferimento per le operazioni di configurazione che è necessario svolgere. Naturalmente, il primo passo necessario è l'installazione del sistema operativo che, come già anticipato, sarà sostituito da una versione più recente di quella prevista di default; similmente, anche ROS Melodic Morenia sarà inizialmente sostituito dalla sua versione successiva: ROS Noetic Ninjemys. MAVROS è invece il protocollo di comunicazione adottato per l'interfaccia tra companion computer e flight director computer, ma, come successivamente spiegato, anche questo verrà sostituito da un differente protocollo:  $\mu$ XRCE-DDS. L'adozione di un diverso protocollo dipende da un'altra esigenza, cioè quella di voler sostituire completamente ROS con il più nuovo ROS2, ma anche su questo punto chiarimenti verranno forniti in seguito. Il driver della fotocamera è poi indispensabile per garantire il funzionamento di questo componente e la sua interfaccia con il companion computer attraverso ROS. Infine, PX4 Avoidance non verrà implementato con la configurazione personalizzata qui sviluppata; questo dipende principalmente da problemi di compatibilità e dal fatto che,

come anche sottolineato dalla documentazione ufficiale, questa funzione al momento non è più supportata da PX4.

### 2.3.1 ROS <sup>[25][26]</sup>

Il Robot Operating System è un framework comprensivo di librerie e strumenti utile alla scrittura, programmazione ed esecuzione di software ad uso robotico. Esso è un meta sistema operativo per robot perché è in grado di espletare alcune funzioni tipiche di un sistema operativo, quali l'implementazione di funzioni ad uso comune, lo scambio di messaggi tra processi, il controllo dei dispositivi a basso livello e la gestione di pacchetti.

Tra i principali vantaggi di ROS è importante citare il fatto che sia open source e con un'ampia e attiva community di supporto; garantisce inoltre la compatibilità con un numero molto ampio di librerie e dispositivi di terze parti, risultando quindi particolarmente vantaggioso per quelle applicazioni che, come robot e droni, richiedono un efficace interfacciamento e coordinazione di molti elementi distinti e intercambiabili, quali sensori e attuatori di diverso genere.

Un altro grande vantaggio è la modularità di ROS: questo si basa in fatti sulla suddivisione dei singoli processi in unità modulari chiamate nodi; un nodo è quindi un singolo processo che comunica con gli altri nodi attraverso modalità che saranno a breve descritte. Lo sviluppo software di un robot è infatti molto complesso proprio perché, come già detto, è necessaria un'efficace gestione simultanea di molti elementi hardware e dei relativi processi; proprio per questo motivo, realizzare il software sfruttando un'architettura altamente modulare è utile, non solo per semplificare la fase di scrittura e test del codice, ma permette poi di effettuare con grande agilità interventi di modifica e personalizzazione anche qualora la configurazione hardware dovesse essere modificata. Si precisa poi che ciascun nodo è sempre contenuto all'interno di un pacchetto, insieme alla definizione dei messaggi che il nodo utilizza ed eventualmente anche con altri nodi; infatti, ogni nodo deve essere contenuto in un pacchetto, ma lo stesso pacchetto può contenere anche più di un solo nodo.

La comunicazione tra nodi può avvenire secondo tre diverse modalità: topic, service e actions. Ciò che accomuna i tre modi di comunicare è lo scambio di messaggi: i messaggi contengono le informazioni che si necessita di scambiare ordinate secondo una struttura prestabilita e specifica del tipo di messaggio; a seconda del tipo di messaggio questo potrà contenere uno o più numeri, una

posizione, una stringa o qualsiasi tipo di informazione organizzata secondo un'architettura che può essere standard o personalizzata.

I ROS topic prevedono lo scambio di informazioni tra un nodo publisher, che pubblica un messaggio sul topic con una certa frequenza, e uno o più nodi subscriber che lo ricevono. Questa modalità è tipica di un sensore che trasmette con continuità delle letture che un'unità di processamento potrebbe essere interessata a ricevere, si pensi ad esempio ad un sensore di posizione.

I ROS services prevedono un nodo server che risponde alle richieste dei nodi client; i messaggi inviati in questo caso sono quindi due: una richiesta dal client verso il server e una risposta dal server verso il client. Un esempio semplice potrebbe essere un service "somma di due numeri": il nodo client dovrà inviare al server un messaggio contenente i numeri da sommare e il server risponderà con il risultato dell'operazione.

Le ROS actions sono a prima vista simili ai services, tanto più che si compongono di topic e services. Anche in questo caso prevedono una richiesta da parte di un client e una risposta da parte di un server ma, a differenza dei services, il loro utilizzo è previsto per processi di più lunga esecuzione. Infatti, in questo caso il server fornisce al client, attraverso un topic, un feedback con continuità per tutta la durata del processo ed è prevista la possibilità da parte del client di annullare la richiesta. La struttura della action prevede, come già detto, un topic di feedback, ma anche un service (detto goal) che il client utilizza per chiamare il server e attivare la action e un service (detto result) che il server utilizza per comunicare al client l'esito del processo. Una action è utile, ad esempio, quando si richiede al robot di raggiungere una certa posizione: tramite il goal viene trasmessa la posizione finale da raggiungere, mentre tramite il result si conferma il raggiungimento della posizione al termine del processo. L'utilizzo della action è giustificato dal fatto che questa operazione può richiedere un tempo anche lungo, con la necessità di ricevere, attraverso il topic di feedback, informazioni sulle posizioni intermedie attraversate e di disporre della possibilità di interromperne l'esecuzione qualora dovessero essere riscontrate anomalie.

Un utile strumento di ROS è `ros2 bag`<sup>[27]</sup>, grazie al quale è possibile registrare i dati pubblicati sui topic del sistema per salvarli all'interno di un database; è a discrezione dell'utente la scelta di quanti e quali topic registrare. Di fatto, dopo aver chiamato lo strumento mediante l'apposita linea di comando da terminale, `ros2 bag` assumerà il ruolo di subscriber per quei topic che l'utente ha indicato con il comando. Lo scopo di ciò è evidentemente la possibilità di avere accesso ai dati di un progetto ROS anche in un secondo momento, per poter così effettuare un replay od altre analisi. Emerge allora chiaramente

l'utilità di questo strumento sia per l'identificazione di eventuali errori o problematiche che potrebbero sorgere all'interno dell'ambiente ROS, ma ancor più per poter analizzare in fase di post processing i dati raccolti ed elaborati durante l'attività.

L'ultimo importante elemento di ROS che è utile introdurre è il launch file. Un launch file consente, quando eseguito, di avviare simultaneamente molteplici nodi e, naturalmente, con le impostazioni ed i parametri desiderati. L'utilità dei launch file sta nel fatto che, con un solo comando, è possibile avviare un intero progetto, i cui nodi andrebbero altrimenti lanciati uno ad uno e ciascuno su di una diversa finestra del terminale. Infine, un launch file consente anche di creare direttamente un bag file senza che l'utente debba farlo manualmente ogni volta. In sintesi, grazie ad un bag file è possibile avviare con un solo comando un intero progetto e registrare anche i dati che questo produce.

Dopo aver compreso che cos'è ROS, come funziona e a cosa serve, è necessario stabilire quale versione di questo utile strumento installare a bordo del drone. Infatti, a partire dal 2007, anno di nascita di ROS, molte versioni sono state distribuite: in particolare a maggio di ogni anno una nuova versione viene rilasciata; ogni due anni viene rilasciata una versione LTS (long term support) supportata per cinque anni, a differenza delle versioni normali che sono supportate solo per due anni.

ROS venne inizialmente sviluppato dallo Stanford Artificial Intelligence Laboratory con scopi accademici e non commerciali; per questo, fattori quali la sicurezza, la topologia delle reti o l'uptime non erano stati tenuti in particolare considerazione. Inoltre, il funzionamento di ROS dipende da un unico nodo detto ROS master responsabile della gestione della comunicazione tra gli altri nodi e ciò questo rappresenta evidentemente un single point of failure, che può rivelarsi potenzialmente invalidante soprattutto per alcune applicazioni. Dall'esigenza di superare questi limiti nasce ROS 2, sviluppato sulla base dell'esperienza maturata negli anni con ROS e con l'obiettivo di fornire uno strumento sfruttabile a livello commerciale: per limitarsi ad alcuni esempi, ROS 2 non necessita di un nodo master, adotta protocolli di comunicazione sicuri, supporta l'esecuzione real-time e offre una serie di altri vantaggi rispetto al suo predecessore.

Dati i vantaggi di ROS 2 rispetto a ROS, appare naturale la scelta di impiegare il sistema più recente dei due per assicurare stabilità e compatibilità con possibili applicazioni future, oltre che con la più recente versione del firmware dell'autopilota ed il suo protocollo di comunicazione  $\mu$ XRCE-DDS. Di fatto però alcune funzioni sviluppate per PX4 in passato sono pensate per operare con ROS e non sono sempre compatibili con ROS 2. Con l'obiettivo di massimizzare

la flessibilità e la compatibilità della configurazione in corso di realizzazione, si è quindi scelto, almeno inizialmente, di installare a bordo del companion computer sia ROS, nella versione “Noetic Ninjemys”, che ROS 2 nella versione “Foxy Fitzroy”. Si noti che per quanto riguarda ROS 2 la versione scelta non è affatto la più recente a disposizione, addirittura, la versione “Foxy” da giugno 2023 non risulta neanche più supportata, si veda il paragrafo 2.3.2 relativo al sistema operativo per una spiegazione in merito a tale scelta.

È necessario precisare che la compresenza di ROS e ROS 2 sullo stesso dispositivo, se non opportunamente gestita, potrebbe causare problemi di interferenza. All’avvio di un computer su cui ROS è installato, viene eseguito un file di setup, o bashrc file, che garantisce l’accesso ai comandi di ROS. Le istruzioni contenute in questo file devono essere opportunamente verificate per accertarsi che la versione di ROS richiamata all’avvio sia quella che l’utente desidera; nello specifico, l’utente deve decidere se ROS o ROS 2 deve essere attivato di default all’avvio. Qualora si desiderasse cambiare temporaneamente quest’impostazione per adoperare la versione di ROS non attiva di default, è sufficiente richiamare da terminale il corrispondente bash file, ma sulla finestra del terminale verrà visualizzato un messaggio di errore che informa della disattivazione della versione di ROS precedentemente attiva. È possibile che tale errore sia segnalato all’apertura di ogni finestra del terminale, ma ciò non costituisce un problema purché il testo del messaggio informi della disattivazione della versione di ROS che non si intende utilizzare. In ogni caso, la causa di ciò è da ricercarsi all’interno del file bashrc che potrebbe accidentalmente contenere le linee di comando che richiamano entrambe le versioni; la versione attiva sarà la seconda ad essere richiamata ed il messaggio di errore riguarderà quindi la disattivazione dell’altra.

### 2.3.2 Sistema operativo <sup>[28][29]</sup>

Il sistema operativo che offre la maggiore flessibilità oltre che la migliore compatibilità con ROS e con i protocolli di interfaccia con l’autopilota è Linux Ubuntu; per questo motivo esso è risultato la scelta più opportuna per lo sviluppo di questo progetto. Inoltre, si tratta anche del sistema fornito di default con il supporto USB del kit, ciò a garanzia della sua piena compatibilità con PX4 e con il companion computer.

Per quanto riguarda la scelta della versione da impiegare, la decisione più sensata potrebbe sembrare quella di installare la più recente versione stabile disponibile. Tuttavia, si è in precedenza fatta notare l’esigenza di installare sul



companion computer sia ROS che ROS 2 ed è quindi opportuno effettuare delle valutazioni di compatibilità prima di procedere con una scelta definitiva.

L'installazione della più recente versione LTS di ROS (Noetic Ninjemys) è suggerita in combinazione con la versione 20.04 di Linux: scegliendo una versione del sistema operativo più recente si correrebbe quindi il rischio di incappare in incompatibilità o malfunzionamenti di ROS, cosa che è naturalmente di interesse evitare. Similmente, per quanto riguarda ROS 2, l'installazione delle più recenti versioni LTS è suggerita con versioni di Ubuntu successive alla 20.04; per questo motivo si è scelto di adoperare ROS 2 Foxy Fitzroy il cui utilizzo è raccomandato in combinazione con la versione in questione del sistema operativo.

In sintesi, la scelta della versione di ROS, ROS 2 e del sistema operativo dipende dall'esigenza di voler adottare in ogni caso lo strumento più recente disponibile, ma con il vincolo di garantire sempre la perfetta compatibilità tra i vari elementi.

Una volta installato il sistema operativo, un'importante operazione di configurazione che deve essere eseguita è l'abilitazione della connessione con SSH (secure shell protocol). Infatti, quando il drone è a terra, è possibile accedere al companion computer attraverso delle periferiche ad esso collegate, ma ciò non è possibile durante il volo, rendendo necessaria la ricerca di un'alternativa. La connessione SSH permette di risolvere il problema, perché consente di accedere al terminale del companion computer attraverso un normale computer desktop, purché i due siano connessi alla stessa rete. Perché questo sistema funzioni è necessario un semplice e rapido intervento di configurazione all'interno del sistema operativo del companion computer e l'installazione, sul computer desktop, di un software apposito. A questo punto, lanciando il software in questione e digitando l'indirizzo di rete del companion computer, si potrà aprire una finestra del terminale di questo secondo mediante la quale, completata l'autenticazione dell'utente, sarà possibile operare liberamente.

### 2.3.3 MAVROS<sup>[30]</sup>, MAVLink<sup>[31]</sup> e $\mu$ XRCE-DDS<sup>[32][33]</sup>

Come già anticipato, MAVLink e  $\mu$ XRCE-DDS sono due protocolli di comunicazione necessari per interfacciare il flight director computer con il companion computer. Di fatto è stata già descritta l'architettura dell'autopilota ed è bene ricordare che è il suo middleware a permettere le comunicazioni sia internamente che verso l'esterno grazie a dei topic uORB. Per quanto riguarda il

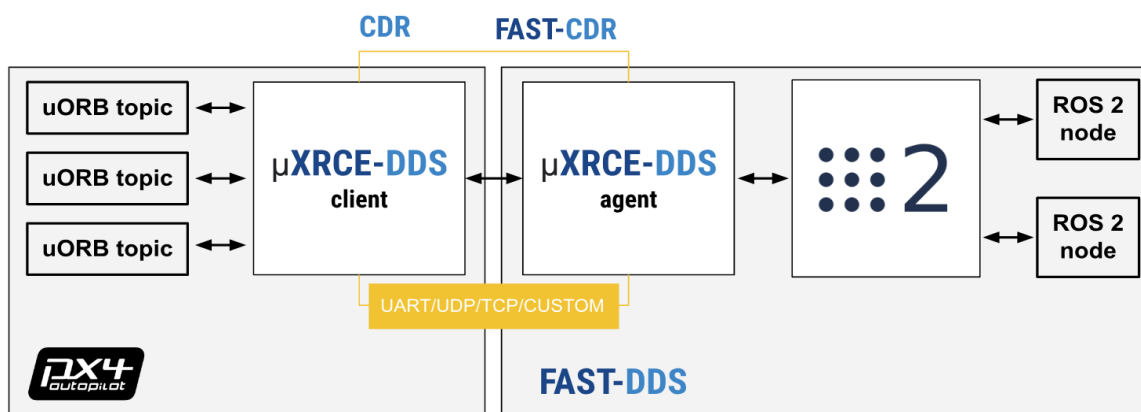
companion computer invece, come appena spiegato, sarà ROS, attraverso i suoi protocolli di comunicazione quali topic o service, a gestire le informazioni ed il loro processamento. Risulta quindi chiara l'esigenza di introdurre un elemento che possa, da un lato comunicare con il middleware di PX4, e cioè con uORB, e dall'altro con ROS sul companion computer. Entrambi MAVLink e  $\mu$ XRCE-DDS sono utili allo scopo ed ambedue sono supportati da PX4, la differenza principale sta nel fatto che il secondo è strutturato per comunicare più agilmente utilizzando ROS2 sul companion computer.

Per cominciare, MAVLink è un protocollo per lo scambio di messaggi particolarmente leggero e quindi adatto sia alla comunicazione con il drone che tra i suoi componenti a bordo. MAVLink adotta un particolare paradigma ibrido, nel senso che i dati vengono gestiti da topic publish-subscribe, mentre i protocolli di configurazione come il mission protocol o il parameter protocol sono trasmessi con sistema point-to-point. Le principali caratteristiche di MAVLink sono la sua straordinaria leggerezza, che lo rende adeguato a comunicazioni anche con una larghezza di banda particolarmente limitata; l'affidabilità, infatti, grazie all'implementazione di metodi di diagnostica degli errori, consente una comunicazione robusta anche in caso di elevata latenza o rumore sul canale; è compatibile con un ampio spettro di linguaggi di programmazione e sistemi operativi; consente la connessione contemporanea di un numero di dispositivi fino a 255; infine permette sia la comunicazione onboard tra gli elementi del drone, che offboard, cioè con la ground station per esempio. A tale proposito si sottolinea che è proprio grazie a MAVLink che il radio comando viene connesso con il drone.

Il middleware di PX4 è, di default, già configurato per comunicare con MAVLink, è dal lato di ROS, sul companion computer, che è necessario assicurarsi che sia installato MAVROS. MAVROS è un pacchetto ROS contenente tre nodi, incluso uno chiamato proprio `mavros_node`. La funzione di questo nodo è quella di permettere il passaggio di messaggi da ROS a MAVLink e viceversa; funge in pratica da traduttore, garantendo a ROS la possibilità di comunicare con MAVLink e quindi con l'autopilota.

Dopo aver installato sul companion computer il sistema operativo e ROS, per replicare la configurazione del driver USB fornito di fabbrica, sarebbe quindi necessario installare MAVROS, abilitare la comunicazione tramite protocollo MAVLink e quindi verificare che ROS e autopilota siano in grado di comunicare efficacemente. In pratica però, è già stata evidenziata l'esigenza di affidarsi al più recente ROS 2, con alcune importanti differenze rispetto a ROS: il pacchetto MAVROS non è infatti disponibile per ROS 2, il cui impiego è invece previsto unitamente al protocollo  $\mu$ XRCE-DDS.

Il protocollo  $\mu$ XRCE-DDS è una versione alleggerita di DDS, l'acronimo infatti significa eXtremely Resource Constrained Environment - Data Distribution Service. Funziona con un principio molto simile sia a quello adottato dal bus messaggi del middleware di PX4 che da ROS, permette infatti una comunicazione efficiente e decentralizzata di più componenti o nodi, ricorrendo all'uso di topic con accesso di tipo publisher/subscriber. In pratica serve a collegare un ambiente con risorse limitate, nel caso in esame l'autopilota, con un DDS globale, in questo caso ROS 2; lo scopo viene raggiunto grazie all'impiego di un'architettura client-server che prevede la connessione dei dispositivi con risorse limitate, detti *XRCE Clients*, con il server, chiamato *XRCE Agent*, incaricato di rispondere in vece dei dispositivi Client all'interno del DDS globale. Con l'intento di massima chiarezza, l'architettura è riportata anche in figura:



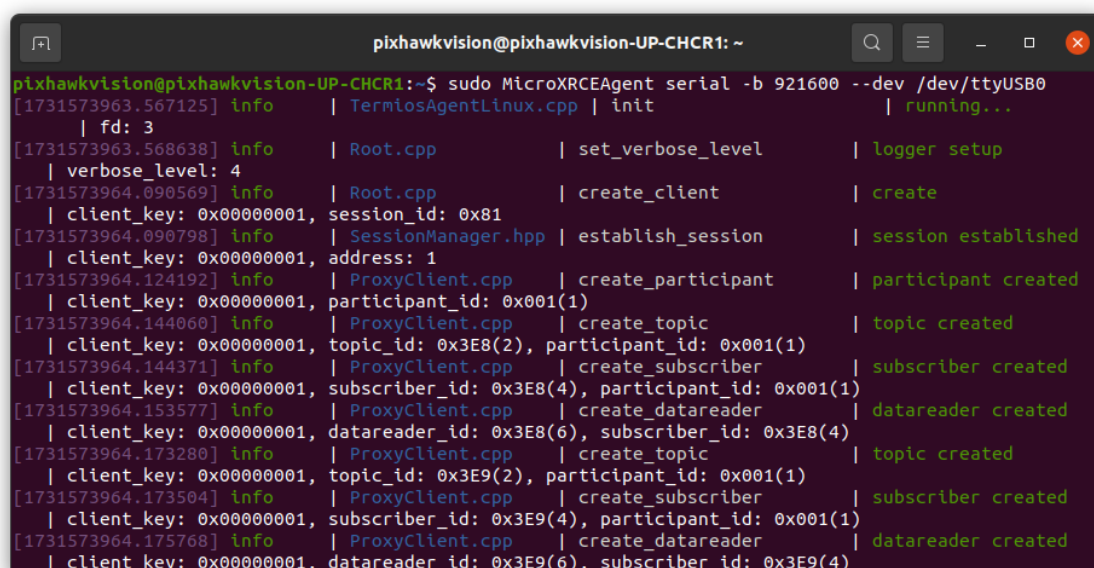
**Figura 23:** Schema di funzionamento del middleware  $\mu$ XRCE-DDS con PX4 e ROS 2

Il grande vantaggio dato dall'impiego di  $\mu$ XRCE-DDS sta essenzialmente nel fatto che, come spiegato, la sua logica di funzionamento è pressoché identica sia a quella del middleware di PX4 che a quella di ROS 2. In maniera molto più snella rispetto a MAVLink, che richiedeva una doppia traduzione,  $\mu$ XRCE-DDS permette al companion computer di interfacciarsi con i topic uORB come se fossero nodi di ROS 2 e, allo stesso modo, il middleware di PX4 può interagire con i nodi di ROS 2 come se fossero topic uORB. Naturalmente, questo protocollo presenta anche altri vantaggi: essendo nato specificatamente per i microcontrollori, è stato progettato per minimizzare il consumo di risorse ed in particolare l'impatto sulla memoria; infine, similmente a MAVLink, è compatibile con un ampio numero di sistemi operativi.

Per configurare il drone in modo che i due computer di bordo comunichino tramite il protocollo  $\mu$ XRCE-DDS, è necessario eseguire alcune operazioni su entrambi. Per quanto riguarda il companion computer, una volta installato ROS2, si deve procedere installando l'agent di  $\mu$ XRCE-DDS e verificare che

questo sia configurato per comunicare attraverso la porta esterna cui è fisicamente connesso via cavo il flight controller, in questo caso specifico la porta TELEM2. Per quanto riguarda l'autopilota, se questo è aggiornato alla versione 1.14 è pronto per utilizzare  $\mu$ XRCE-DDS senza dover ricorrere ad ulteriori aggiornamenti od installazioni; ciò che è però necessario fare è modificare alcuni parametri da QGroundControl per disattivare MAVLink e abilitare  $\mu$ XRCE-DDS attraverso la medesima porta TELEM2. Fatto ciò, all'accensione del flight controller, l'autopilota attiverà automaticamente il client di  $\mu$ XRCE-DDS, così che, una volta lanciato l'agent sul companion computer, sarà finalmente consentito lo scambio di messaggi tra ROS 2 e PX4.

Nell'immagine (figura 24) viene mostrata la finestra del terminale sul companion computer dopo l'avvio dell'agent  $\mu$ XRCE-DDS, una volta che questo viene lanciato vengono visualizzati una serie di messaggi che informano l'utente della buona riuscita dell'operazione; naturalmente ciò dipende anche dal fatto che il computer dell'autopilota sia alimentato e quindi in funzione.



```
pixhawkvision@pixhawkvision-UP-CHCR1: ~  
pixhawkvision@pixhawkvision-UP-CHCR1:~$ sudo MicroXRCEAgent serial -b 921600 --dev /dev/ttyUSB0  
[1731573963.567125] info | TermiosAgentLinux.cpp | init | running...  
| fd: 3  
[1731573963.568638] info | Root.cpp | set_verbose_level | logger setup  
| verbose_level: 4  
[1731573964.090569] info | Root.cpp | create_client | create  
| client_key: 0x00000001, session_id: 0x81  
[1731573964.090798] info | SessionManager.hpp | establish_session | session established  
| client_key: 0x00000001, address: 1  
[1731573964.124192] info | ProxyClient.cpp | create_participant | participant created  
| client_key: 0x00000001, participant_id: 0x001(1)  
[1731573964.144060] info | ProxyClient.cpp | create_topic | topic created  
| client_key: 0x00000001, topic_id: 0x3E8(2), participant_id: 0x001(1)  
[1731573964.144371] info | ProxyClient.cpp | create_subscriber | subscriber created  
| client_key: 0x00000001, subscriber_id: 0x3E8(4), participant_id: 0x001(1)  
[1731573964.153577] info | ProxyClient.cpp | create_datareader | datareader created  
| client_key: 0x00000001, datareader_id: 0x3E8(6), subscriber_id: 0x3E8(4)  
[1731573964.173280] info | ProxyClient.cpp | create_topic | topic created  
| client_key: 0x00000001, topic_id: 0x3E9(2), participant_id: 0x001(1)  
[1731573964.173504] info | ProxyClient.cpp | create_subscriber | subscriber created  
| client_key: 0x00000001, subscriber_id: 0x3E9(4), participant_id: 0x001(1)  
[1731573964.175768] info | ProxyClient.cpp | create_datareader | datareader created  
| client_key: 0x00000001, datareader_id: 0x3E9(6), subscriber_id: 0x3E9(4)
```

**Figura 24:** Avvio del client  $\mu$ XRCE-DDS dal companion computer

### 2.3.4 Camera

L'installazione di un companion computer, con sistema operativo Linux Ubuntu, a bordo del drone ha lo scopo di garantire un ampio spettro di compatibilità e connettività con sensori e dispositivi a cominciare dalle camere. Infatti, il flight controller supporta solo un limitato numero di equipaggiamenti preconfigurati anche se con un sostanziale vantaggio; i computer che supportano l'autopilota

PX4 sono intrinsecamente predisposti ad interfacciarsi, ad esempio, con moduli GNSS compatibili o sensori optical flow. Questo implica che una modifica dell'hardware connesso al flight controller richiede interventi di configurazione software molto rapidi, quali la modifica di alcuni parametri tramite QGroundControl, o potrebbe persino non richiedere alcun intervento. L'utilizzo di equipaggiamenti connessi al companion computer richiede invece una diversa attenzione e l'installazione di software dedicato.

Per poter utilizzare la camera in dotazione al PX4 Vision Autonomy Development Kit è allora necessario, come primo passo, installare sul companion computer il driver specifico della fotocamera, ciò allo scopo di permettere la comunicazione tra il computer e la camera stessa. Il driver è infatti il software che permette ad un sistema operativo di interfacciarsi con un dispositivo hardware esterno e deve quindi essere compatibile con il sistema operativo stesso.

Il driver è solitamente fornito dalla stessa azienda produttrice del dispositivo, ma la sua installazione, sebbene indispensabile, non è sufficiente per permettere un efficace sfruttamento della camera in simbiosi con gli altri elementi del drone. A tal proposito è ROS lo strumento impiegato per garantire l'elaborazione e condivisione dei dati tra i diversi equipaggiamenti.

Come spiegato in precedenza, ROS si interfaccia con l'autopilota attraverso appositi protocolli di comunicazione e sarà quindi lo stesso ROS a dover raccogliere i dati della camera, elaborarli e sfruttarli per l'implementazione di applicazioni. È quindi necessario installare un pacchetto ROS i cui nodi sono responsabili della comunicazione con la camera e la pubblicazione dei dati che essa produce su appositi topic. Questa applicazione pratica permette di apprezzare in particolar modo l'efficacia dell'architettura modulare di ROS: i nodi di interfaccia con la camera ne rendono disponibili i dati, altri nodi potranno, con ruolo di subscriber, avere accesso alle informazioni, elaborarle e produrre quindi altri dati da pubblicare su altri topic e così via. Alla fine della catena è possibile semplicemente raccogliere i dati su un file oppure, per applicazioni legate al volo, inviarli all'autopilota grazie ai nodi di interfaccia con  $\mu$ XRCE-DDS.

Dopo questo utile chiarimento sull'architettura del software necessario al buon funzionamento e proficuo sfruttamento della camera, è possibile discutere della sua implementazione, ma a questo proposito sorgono alcune criticità. La camera Occipital Structure Core non è tra le più ampiamente diffuse sul mercato e non è quindi universalmente supportata. Per quanto riguarda il driver è possibile reperirne una versione compatibile con Linux ubuntu 20.04, sebbene non con facilità per via del fatto che quella qui impiegata non è la versione più

recente di questo sistema operativo. La maggiore difficoltà sussiste però nell'impossibilità di reperire un pacchetto ROS compilabile e funzionante che sia compatibile con questo modello di camera e con ROS 2 Foxy. La scelta della versione di ROS ha infatti contribuito all'insorgere della problematica, ciò perché ROS 2 non gode dell'eredità del suo predecessore ROS che ancora vanta l'appoggio di una nutrita community, ma soprattutto perché si tratta di una versione di ROS 2 oggi superata da altre più recenti. È in parte anche a causa di queste difficoltà che si è giunti alla decisione di impiegare nella fase di test una fotocamera differente rispetto a quella già in dotazione al drone, ma questo argomento sarà oggetto di più approfondite argomentazioni nella sezione dedicata al setup per i voli di prova.

## 3) Setup sperimentale

In richiamo a quanto già accennato nelle sezioni introduttive, oltre alla configurazione del drone, l'ulteriore obiettivo inizialmente posto è quello di completare alcuni voli di prova che possano in primo luogo permettere di testare e validare la configurazione software, il suo buon funzionamento e l'armoniosa integrazione di tutti i sensori e componenti hardware. Successivamente lo scopo dei voli di prova è quello di testare l'efficacia e la precisione di diversi sistemi per la determinazione della posizione del veicolo tramite lo sfruttamento di diverse tecnologie a potenziamento delle reti satellitari GNSS, per poterli ordinare in un benchmark in base alle prestazioni.

### 3.1 Modifiche hardware

Uno dei principali vantaggi del PX4 Vision Autonomy Development Kit è la possibilità, da parte dell'utente, di poter intervenire con modifiche hardware sostituendo o semplicemente aggiungendo sensori ed equipaggiamenti. La flessibilità che il veicolo offre è una caratteristica particolarmente apprezzabile che consente una certa versatilità ed adattabilità al tipo di missione per cui si intende impiegare la piattaforma in questione.

Seguendo l'obiettivo di voler sviluppare una configurazione personalizzata ottimale, appare quasi come un'esigenza la possibilità di sfruttare la flessibilità del kit per apportare alcune utili e pratiche modifiche.

Nel caso in esame, la configurazione hardware che si intende implementare per lo svolgimento dei test di cui ai capitoli successivi, si è reputato opportuno operare due principali variazioni rispetto al setup di fabbrica: la sostituzione del modulo GPS con un modello migliore che supporta più funzionalità e l'aggiunta di una diversa camera da connettere al companion computer. Maggiori dettagli su queste scelte e sui nuovi componenti installati sono quindi esposti di seguito.

#### 3.1.1 Modulo GNSS

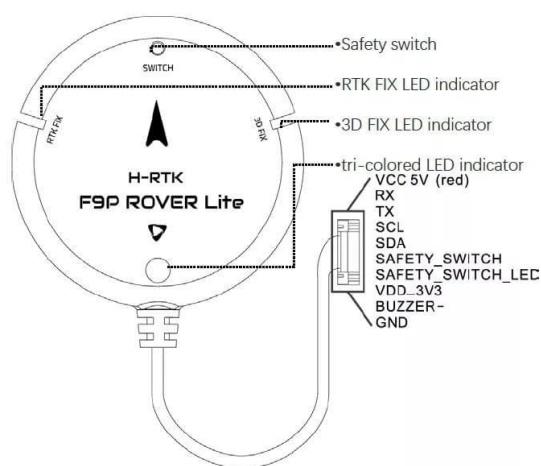
Siccome l'attività sperimentale si incentrerà sulla valutazione dei diversi sistemi per la determinazione della posizione mediante GPS e GPS aumentato, è



fondamentale che il drone sia equipaggiato con un modulo GNSS che possa garantire le migliori prestazioni possibili. Il componente previsto nella dotazione di fabbrica del PX4 Vision Autonomy Development Kit è il Holybro M8N GPS<sup>[34]</sup>. Il M8N, che così chiameremo d'ora in avanti per brevità, è capace di assolvere il proprio compito con buona efficacia ed è compatto, ma purtroppo non è compatibile con i sistemi RTK, tecnologia che si intende però testare e a proposito della quale saranno in seguito fornite spiegazioni dettagliate. Questo equipaggiamento inoltre, non è certo il più preciso e sofisticato tra quelli disponibili e non rappresenta quindi il top di gamma.

Dall'esigenza di voler utilizzare un modulo GNSS che sia più moderno e che offra migliori prestazioni, si è quindi deciso di sostituire il componente di fabbrica con il suo alter ego migliore e più recente: il Holybro H-RTK F9P GNSS <sup>[35]</sup>, F9P per brevità. La scelta di questo modello dipende dal fatto che questo sia già in dotazione al gruppo di ricerca, offra ottime prestazioni, inclusa la compatibilità con la tecnologia RTK, ed infine dal fatto che, soprattutto in forza della sua somiglianza con il M8N, sia pienamente compatibile con l'autopilota del drone. Più in particolare, a differenza del M8N, il F9P è multibanda, questo significa che il dispositivo consente di ottenere prestazioni migliori, maggiore affidabilità e minore tempo di convergenza con l'uso di RTK. Supporta poi uno spettro significativamente più ampio di sistemi satellitari (per esempio il Galileo che il M8N non supporta) ed ha un minore tempo di update, particolarmente apprezzabile per applicazioni dinamiche con precisioni fino al centimetro.

Si fornisce quindi un'immagine con una rappresentazione schematica del componente installato (figura 25).



SKU12017 H-RTK F9P RoverLite

**Figura 25:** Modulo GNSS F9P installato sul veicolo in sostituzione del M8N

Per completezza è anche importante evidenziare le criticità che il diverso equipaggiamento pone: anzitutto il modulo F9P è più costoso rispetto a quello di fabbrica, ma essendo comunque già in dotazione al gruppo di ricerca, questo non rappresenta una grave problematica. L'aspetto che invece deve essere considerato con la dovuta attenzione riguarda il fatto che questo componente è più ingombrante e più pesante rispetto a quello al posto del quale viene installato; è dunque buona cosa valutarne l'impatto sulla posizione del baricentro e sulla stabilità del veicolo, ma su questo aspetto tornerà il paragrafo 3.1.2 che segue.

### **3.1.2 Camera**

Non intendendo implementare applicazioni di obstacle avoidance o simili, potrebbe non essere immediatamente chiaro il ruolo svolto dalla camera al fine di testare i sistemi di posizionamento satellitare. Di fatto però, proprio per rendere più rigorose le misurazioni che verranno effettuate, è necessario poter valutare la posizione del drone nello spazio rispetto ad un riferimento fisso. Per rispondere a questa esigenza, si è deciso di introdurre nella zona di volo un marker, di modo che, tracciandone la posizione all'interno del campo visivo della camera, si potesse in ogni istante ricostruire la posizione effettiva del veicolo. Una descrizione più approfondita di questa tecnologia e del modo in cui può essere sfruttata sarà fornita in seguito, per il momento ciò che è più importante notare è che il drone dovrà essere dotato di una camera orientata nella direzione del marker in ogni istante durante il volo. Per fornire efficacemente una posizione di riferimento e per essere sempre visibile al drone, il marker dovrà essere posizionato al suolo e sarà quindi verso il basso che la camera andrà orientata.

Nel paragrafo relativo alla configurazione software della camera è già emersa l'esigenza di impiegare un dispositivo che possa efficacemente essere interfacciato con il sistema operativo e con ROS 2 Foxy, ma ora si evidenzia la necessità di prendere in considerazione un altro importante requisito: quello relativo alla direzione di puntamento. La camera Occipital Structure Core, con cui il PX4 Vision Autonomy Development Kit è equipaggiato, è montata su di un supporto rigido che ne vincola l'orientamento esclusivamente alla direzione frontale; la problematica si presta quindi a due possibili soluzioni: spostare la camera già presente sul veicolo od installarne una seconda. In ogni caso, per poter equipaggiare sul veicolo una camera rivolta verso il basso senza comprometterne la capacità di volare e la libertà di movimento, la zona più appropriata per ospitare il sensore necessario risulta essere l'area libera sul

retro della carrier plate. A valle di questa scelta emerge l'esigenza di ricorrere ad un dispositivo che sia il più compatto possibile, per evitare di interferire con i supporti dei motori, ma anche a causa della limitata lunghezza del cavo preposto al collegamento della camera con il companion computer. Questa posizione risulta poi particolarmente adeguata perché, prevedendo la collocazione di una massa aggiuntiva sul retro del veicolo, permette di controbilanciare il maggiore peso del diverso modulo GNSS, installato invece nella parte frontale.

A rispettare i requisiti posti è la camera Intel RealSense Depth Camera D435i<sup>[36]</sup> già in dotazione al gruppo di ricerca.



**Figura 26:** RealSense Depth Camera D435i da aggiungere al veicolo

Questa camera è di fatto simile alla Occipital Structure Core, già in dotazione al drone, sia dal punto di vista dei componenti che integra ma anche delle funzionalità che supporta. Essa è infatti dotata di una camera che opera nel visibile, un sensore RGB, due camere a infrarossi per la visione stereo di profondità, un processore ed un'unità inerziale.

Con un ingombro di 90 mm × 25 mm × 25 mm, questa camera risulta più compatta rispetto alla Occipital Structure Core, ma è soprattutto la sua ampia diffusione a renderla decisamente più appetibile. Questo sensore così popolare offre infatti una maggiore compatibilità, tanto più perché beneficia dell'appoggio di un'ampia comunità di utenti e sviluppatori, nonché del solido supporto tecnico da parte dell'azienda produttrice.

Per quanto riguarda le specifiche tecniche, la camera opera entro un range ottimale compreso tra tre metri e trenta centimetri, con una distanza minima di funzionamento di ventotto centimetri per la visione di profondità. Per sintesi sono date in forma di elenco le specifiche di dettaglio delle fotocamere:

- Camera di profondità
  - Field of View:  $87^\circ \times 58^\circ$
  - Output resolution: fino a  $1280 \times 720$
  - Accuracy:  $<2\%$  a 2 m
  - Frame rate: fino a 90 fps
  
- Camera RGB
  - Field of View:  $69^\circ \times 42^\circ$
  - Frame resolution:  $1920 \times 1080$
  - Sensor resolution: 2 MP
  - Frame rate: 30 fps

## 3.2 Determinazione della posizione

La determinazione della posizione è una delle funzioni più importanti per il buon funzionamento di un velivolo unmanned. La disponibilità di informazioni precise ed affidabili sulla posizione del veicolo è infatti imprescindibile, non solo per poter espletare la maggior parte delle funzioni, ma anche per garantire la sicurezza dello stesso veicolo. In questa sezione si intende fornire un approfondimento sul modo in cui il drone determina la propria posizione e presentare le varie tecnologie migliorative che si intende in seguito testare.

### 3.2.1 Autopilota PX4 – filtro di Kalman esteso <sup>[37]</sup>

Come anticipato nella sezione dedicata all'autopilota, per la determinazione della posizione, PX4 si avvale di un algoritmo complesso noto come filtro di Kalman esteso, EKF2 per brevità. Per poter comprendere nel dettaglio di cosa si tratta e come funziona, è innanzitutto necessario un chiarimento sulla nomenclatura utilizzata: l'autopilota, infatti, etichetta con diversi nomi le differenti informazioni di posizione di cui dispone.

Con GPS Position sono identificati i dati sulla posizione ottenuti sfruttando il ricevitore GNSS, prima che questi vengano processati dall'autopilota ma già combinati con eventuali informazioni di correzione. I dati sulla posizione così ottenuti permettono quindi di collocare il veicolo all'interno di un sistema di riferimento globale, ma, in assenza di correzioni, sono poco precisi e fortemente

dipendenti da fattori quali il numero di satelliti visibili od altre criticità legate al sistema satellitare.

Con Global Position<sup>[38]</sup> si intendono delle informazioni di posizione che, sebbene in apparenza simili alla GPS Position, presentano in realtà una importante differenza. Ancora una volta esprimono la posizione del veicolo in un sistema di riferimento globale, ma, questa volta, le coordinate vengono generate a valle di un processo di raffinazione che combina diverse fonti di dati, non esclusivamente il ricevitore GNSS. La posizione globale così ottenuta è quindi più precisa rispetto ai dati GPS non processati, ma soprattutto dovrebbe risultare più stabile. Infine, si precisa per completezza che il sistema di riferimento globale adoperato è il WGS84.

Con Local Position<sup>[39]</sup> si intende invece la posizione espressa secondo il sistema di riferimento NED, centrato nella posizione del veicolo quando il filtro di Kalman esteso viene avviato. Infatti, anche questa posizione viene calcolata dal EKF2, combinando i dati messi a disposizione dai vari sensori.

Per quanto riguarda il filtro di Kalman esteso, è bene precisare che esso è contenuto all'interno della libreria ECL (acronimo per Estimation and Control Library) ed è responsabile della stima, a partire dai dati dei sensori, non solo della posizione, ma anche dell'assetto. Sebbene la stima dell'assetto non sia di primario interesse ai fini di questa tesi, essa è, a livello di algoritmo, strettamente correlata alla stima della posizione e non è quindi possibile prescindere dal considerarla.

Venendo quindi al funzionamento del EKF2, esso può operare secondo differenti modalità operative a seconda dei sensori sui quali gli è possibile fare affidamento. In ogni caso, perché l'algoritmo possa funzionare, è necessario che esso disponga almeno dei dati di imbardata, altezza e della piattaforma inerziale: grazie a questi il EKF è in grado di calcolare le rotazioni, la velocità verticale e la posizione verticale. Per calcolare eventuali altri stati, è necessario l'impiego di sensori che forniscano dati aggiuntivi rispetto a quelli minimi richiesti. Segue dunque una panoramica relativa ai dati utili al filtro di Kalman esteso e agli equipaggiamenti che li generano<sup>[37]</sup>.

L'unità inerziale, o IMU dall'acronimo inglese Inertial Measurement Unit, è responsabile della misurazione, rispetto agli assi corpo, delle variazioni degli angoli di assetto e della velocità. Per quanto riguarda i dati di imbardata è invece il magnetometro la fonte più comunemente utilizzata, sebbene possa essere sostituito da un sistema di computer vision che sfrutti una fotocamera. Infine la misura dell'altezza è quella che permette maggiore flessibilità, perché può essere ottenuta sfruttando il GPS, la pressione barometrica, un range finder,

un sistema di computer vision oppure anche una combinazione di questi. Per consentire al EKF di funzionare, le informazioni di imbardata e altezza devono essere fornite con una frequenza minima di 5 Hz, mentre il requisito sale a 100 Hz per i dati della piattaforma inerziale. Quando queste condizioni sono rispettate, il filtro si avvia e completa, prima l'inizializzazione degli stati, e poi una procedura detta tilt and yaw alignment. Solo successivamente il EKF può procedere con la transizione ad una diversa modalità di funzionamento che consenta di sfruttare i dati di un maggiore numero di sensori.

La selezione di eventuali ulteriori sensori da adoperare, così come la scelta della sorgente dei dati di altezza, dipende dalle preferenze espresse dall'utente attraverso specifici parametri dell'autopilota: ad esempio, il parametro EKF2\_BARO\_CTRL serve per abilitare o disabilitare i dati del barometro.

Per quanto riguarda la stima della posizione globale, i dati di partenza sono quelli ottenuti grazie al ricevitore GNSS. I dati satellitari vengono infatti utilizzati per determinare la posizione e la velocità del veicolo, ma solo se si verificano due condizioni: in primis il GPS deve essere abilitato mediante il parametro EKF2\_GPS\_CTRL ed in secundis deve essere superato un test di qualità. Per rendere più lineare la trattazione però, tutte le informazioni di dettaglio relative al GPS sono raggruppate nel paragrafo che segue (3.2.2), cui si invita a fare riferimento.

A completamento dei dati satellitari, l'algoritmo dell'autopilota può, innanzitutto, integrare eventuali informazioni di correzione se si utilizza un sistema GNSS aumentato. Successivamente il filtro di Kalman esteso, attraverso la procedura di sensor fusion, migliora la stima della posizione avvalendosi degli altri sensori di bordo. Affidandosi esclusivamente ai dati satellitari, è possibile che vengano rilevati degli spostamenti non reali dovuti alla bassa precisione del GPS; questo errore può essere compensato, ad esempio, grazie all'uso del sensore optical flow o di un sistema di computer vision, che forniscano all'algoritmo una percezione degli spostamenti realmente compiuti dal veicolo. Naturalmente, anche l'impiego di questi sensori dipende dal fatto che il loro utilizzo sia abilitato mediante i corrispondenti parametri e che i dati da essi forniti rispettino uno standard minimo di qualità. Per informazioni relative a tutti i parametri, sia di abilitazione che di qualità, si invita a fare riferimento alla documentazione online: sarebbe superfluo ai fini della trattazione inserire elenchi di parametri e corrispondenti valori in questa sede.

### 3.2.2 Sistemi GNSS

Il Global Navigation Satellite Systems<sup>[40]</sup>, per brevità GNSS, è il servizio satellitare che fornisce agli utenti informazioni di tempo e posizione. Nella fattispecie, un ricevitore GNSS capta i segnali provenienti dai satelliti e li utilizza per determinare la propria posizione: ogni satellite emette due onde portanti sinusoidali di diversa lunghezza (L1 ed L2) modulate secondo un codice generato da un algoritmo, che si ripete periodicamente nel tempo. Questo codice, chiamato Pseudo-Random Number (PRN), viene generato contemporaneamente anche dal ricevitore e quindi, misurando lo sfasamento tra il codice ricevuto dal satellite e quello generato internamente, è possibile calcolare la distanza tra ricevitore e satellite. Il segnale inviato da un satellite contiene poi l'informazione sulla sua propria posizione ed allora, captando il segnale di almeno quattro satelliti, è possibile per il ricevitore calcolare la propria posizione nello spazio avvalendosi del principio di triangolazione. Naturalmente le misure così ottenute sono affette da errori di diversa genesi, il cui effetto sulla precisione del calcolo diminuisce con l'aumentare del numero di satelliti in vista del ricevitore. È per questo motivo che è molto apprezzabile la compatibilità del ricevitore Holybro H-RTK F9P GNSS<sup>[35]</sup> con tutte le quattro principali costellazioni: GPS, GLONASS, Galileo e BeiDou, in questo modo il numero di satelliti in vista sarà mediamente molto alto e così la precisione della posizione determinata. A tal proposito è bene tenere in considerazione un secondo parametro, la diluizione della precisione, o DOP dall'inglese Dilution Of Precision, nelle sue varianti HDOP e VDOP, dove H sta per Horizontal e V per Vertical. Essi rappresentano il rapporto tra l'errore nella determinazione della precisione e l'errore della misura e, di fatto, offrono una misura della bontà della disposizione dei satelliti in vista: infatti, la triangolazione non è sensibile solo al numero di satelliti visibili, ma anche alla loro disposizione nello spazio. Se i parametri DOP hanno un valore inferiore a 1, allora ci si trova nella situazione ottimale, quella in cui è possibile raggiungere il massimo livello di precisione.

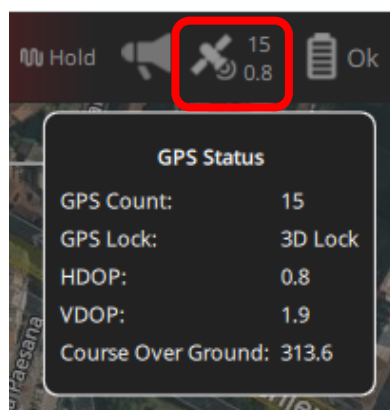
Per quanto riguarda il drone ed il suo autopilota più nello specifico, come già accennato, perché i dati GNSS siano sfruttabili da parte del filtro di Kalman esteso e per la navigazione è necessario che questi superino un test di qualità. Il suddetto test consiste nell'accertare che il GPS funzioni con continuità senza anomalie e rispettando una serie di requisiti per un periodo di tempo definito che l'utente può impostare mediante un parametro (il cui valore di default è 10s).

Più nello specifico, i requisiti da rispettare riguardano<sup>[37][41]</sup>: il numero di satelliti visibili, la deviazione standard dell'errore di posizione orizzontale e verticale, la deviazione standard dell'errore di velocità orizzontale, drift di posizione e quota



e naturalmente la diluizione della precisione. Ognuno di questi requisiti può essere reso più o meno stringente modificando il valore del corrispondente parametro. Le impostazioni di default sono tali da garantire una buona qualità dei dati compatibilmente con le prestazioni di cui i moduli GNSS più diffusi sono capaci; a tal proposito, requisiti più esigenti potrebbero essere troppo stringenti per un ricevitore normale, mentre requisiti troppo rilassati potrebbero non essere sufficienti a garantire la qualità dei dati. Per questo motivo, i parametri dei requisiti non possono essere modificati arbitrariamente, ma solamente all'interno di range definiti.

Quando il veicolo si connette al GNSS, l'utente ne viene informato da un'apposita icona nella status bar di QGroundControl (riquadrata in rosso nella figura 27). Accanto all'icona compaiono poi i due dati prestazionali più rilevanti, cioè il numero di satelliti con cui il ricevitore è connesso il valore del parametro HDOP. Cliccando sulla suddetta icona è possibile far comparire la tendina mostrata in figura per visualizzare ulteriori informazioni.



**Figura 27:** Icona e informazioni sul GPS in QGroundControl

La voce "GPS Lock" informa l'utente del cosiddetto "GPS Fix Type" che descrive il livello di accuratezza e la tipologia di informazioni per la determinazione della posizione ricevute; in sintesi esso descrive la bontà della capacità di determinare la posizione del ricevitore. La dicitura "3D Lock" vuol dire che è possibile determinare latitudine, longitudine ed altitudine del ricevitore e, come già detto, ciò è possibile quando vengono ricevuti i segnali di almeno quattro satelliti. È necessario precisare che, per applicazioni che non necessitano di conoscere l'altitudine, sarebbe possibile conoscere le coordinate del ricevitore connettendosi ad un minimo di tre soli satelliti; in tal caso il fix type sarebbe "2D Lock", ma tale condizione non sarebbe idonea per un veicolo volante quale quello in esame. La peggior condizione comunque è rappresentata dal caso "None" in cui il ricevitore GNSS non è in grado di determinare la posizione. Procedendo invece ora in ordine di accuratezza crescente, migliore del caso "3D

Lock” è il “3D DGPS Lock”: in questo caso l’utente viene informato del fatto che la precisione della posizione determinata viene migliorata grazie al dead reckoning; in pratica i dati dei sensori a bordo, quali quelli della piattaforma inerziale, vengono utilizzati per il calcolo della posizione corrente per propagazione dalla posizione precedente e dunque, confrontando questo risultato con i dati GNSS, ottenere così prestazioni migliori; si tratta in pratica del concetto di sensor fusion, a proposito del quale si invita a fare riferimento al paragrafo 3.2.1.

Il “3D DGPS Lock” rappresenta il massimo grado di GPS Fix ottenibile dal veicolo senza il contributo di tecnologie esterne. Nei paragrafi a seguire verranno descritti due metodi per migliorare le prestazioni: RTK e Spin3 GNSS; maggiori informazioni verranno quindi fornite a breve, per il momento l’aspetto su cui focalizzarsi e che queste inviano al ricevitore GNSS dei messaggi di correzione che possono essere sfruttati per il raggiungimento dei gradi superiori di GPS Fix: “3D RTK GPS Lock (float)” e “3D RTK GPS Lock (fixed)”, il significato di questi verrà a breve chiarito.

### 3.2.3 RTK<sup>[42]</sup>

Il GPS RTK, acronimo per Real Time Kinematic, è un sistema di correzione dei segnali GNSS per aumentarne la precisione fino al centimetro. Esso si basa sullo sfruttamento di una base con un’antenna che invia le correzioni ad uno o più rover, o ad un drone come nel caso in esame. Nel paragrafo precedente si è accennato al principio di funzionamento del normale GPS, basato sulla valutazione della modulazione del segnale PRN. Invece, la base RTK misura la fase delle onde portanti ricevute dai satelliti, ignorandone il contenuto informativo; in questo modo, grazie ad una complessa analisi statistica, è possibile migliorare significativamente la precisione del calcolo della distanza rispetto a quanto sia possibile fare con il metodo standard.

È bene precisare che questo metodo consente di raggiungere precisioni molto elevate, dell’ordine del centimetro, ma esclusivamente per quanto riguarda la determinazione della posizione relativa del veicolo rispetto alla base. L’accuratezza nel determinare la posizione globale è a sua volta legata al calcolo della posizione della base, soggetta comunque ad un’incertezza dell’ordine del metro, come per un normale ricevitore GNSS. Risulta dunque chiaro che l’utilizzo del GPS RTK è particolarmente utile per stabilizzare la posizione calcolata dal veicolo e conoscere con precisione i suoi movimenti in un sistema di riferimento locale, ma comunque non fornisce un significativo vantaggio per la determinazione della posizione del veicolo rispetto al WGS84.

L'equipaggiamento utilizzato, mostrato nella figura 28, è il H-RTK F9P Base<sup>[35]</sup>, abbinato naturalmente al sensore F9P appositamente installato sul drone, così come già specificato nel paragrafo 3.1.1.

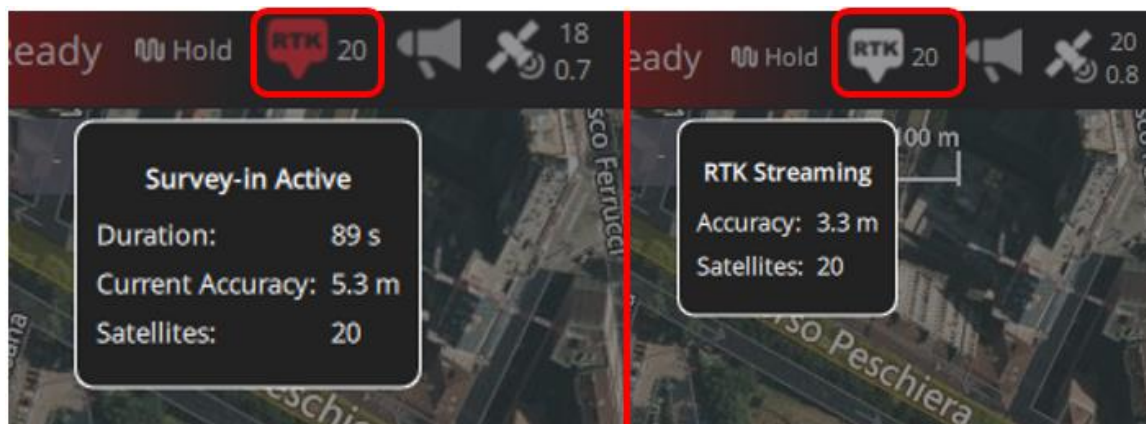


**Figura 28:** Base RTK F9P

La base deve essere collegata al desktop computer che, con funzione di ground station, è anche connesso al veicolo tramite QGroundControl. È infatti attraverso questo software che la base RTK viene configurata per poi inviare i messaggi di correzione. Nel momento in cui la base si connette con QGroundControl ha inizio una fase chiamata survey-in, durante la quale viene determinata la posizione della base. Il survey-in dura solitamente svariati minuti e termina quando vengono raggiunte due condizioni: l'accuratezza della posizione della base deve essere adeguata, con l'incertezza inferiore ad una certa soglia, e deve essere trascorso un tempo minimo; entrambi questi vincoli possono essere modificati dall'utente che, in una pagina dedicata di QGroundControl, può modificare sia il tempo minimo che l'incertezza della posizione agendo sui parametri "survey-in accuracy" e "minimum observation time". In generale questi valori devono essere assegnati in modo da raggiungere un compromesso tra l'esigenza di ottenere la massima accuratezza possibile ed una ragionevole durata della fase di survey-in; bisogna quindi considerare la qualità del segnale satellitare ricevuto ed il numero di satelliti in vista, ovverosia gli aspetti determinanti per il calcolo della posizione della base.

Quando la base viene connessa alla ground station, nella status bar di QGroundControl appare l'icona, riquadrata in rosso nella figura 29 a sinistra, con la scritta RTK in una vignetta rossa. Essa indica che la base è connessa ed

è in corso la fase di survey-in; cliccando sull'icona è possibile visualizzare le informazioni relative allo stato attuale dei parametri relativi a questa fase.



**Figura 29:** Dati RTK su QGroundControl

Al termine del survey-in, ovvero quando i requisiti imposti vengono soddisfatti, avrà allora inizio lo streaming di messaggi di correzione al veicolo; l'utente viene di ciò informato dall'icona RTK bianca, come in figura 29 a destra. Anche in questo caso, con un click sull'icona è possibile visualizzare più informazioni.

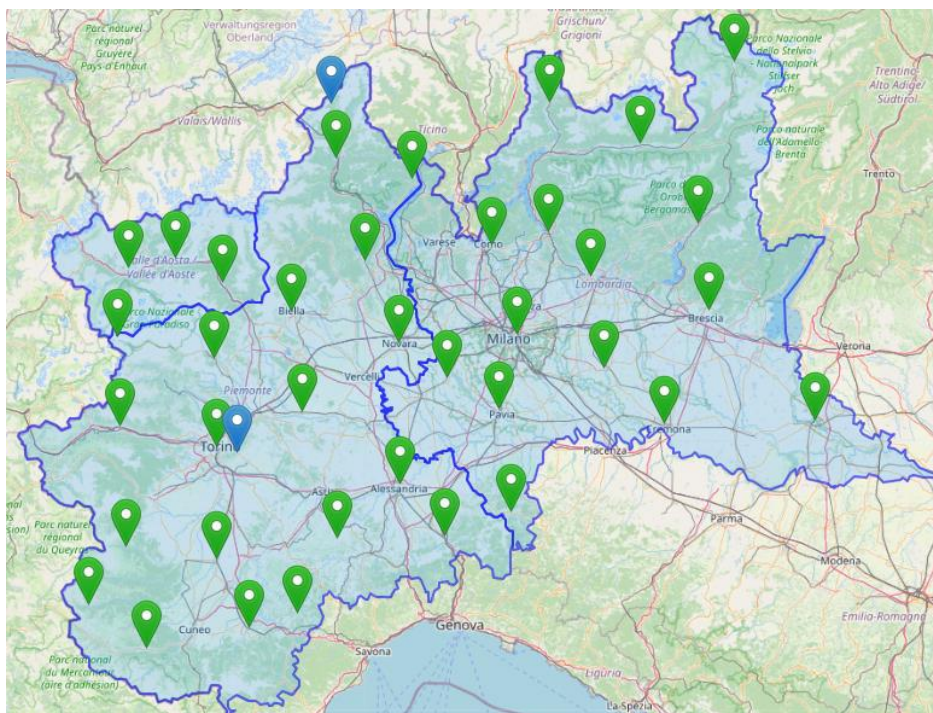
Come anticipato, l'uso di questa tecnologia consente di raggiungere i livelli più alti di GPS fix: 3D RTK GPS Lock (float) e 3D RTK GPS Lock (fixed)<sup>[43]</sup>. Come in precedenza spiegato, il sistema RTK basa il calcolo della posizione sullo sfasamento delle onde portanti: quando il calcolo del numero di fronti d'onda tra satellite e ricevitore porta, come soluzione, ad un numero intero, allora è possibile raggiungere la precisione centimetrica con il grado RTK fixed. Se la soluzione consiste invece in un numero decimale, vuol dire che il calcolo non è ancora giunto a convergenza: le correzioni RTK sono comunque disponibili, ma non è possibile raggiungere la posizione centimetrica: in questo caso il GPS fix è RTK float. Questo secondo caso può verificarsi soprattutto se la ricezione dei segnali satellitari non è buona o se il sistema è stato da poco avviato.

### 3.2.4 Spin3 GNSS<sup>[44][45]</sup>

Spin3 GNSS è un servizio di supporto ai sistemi di posizionamento satellitari che, attraverso l'invio di messaggi di correzione, permette di raggiungere precisioni centimetriche in tempo reale. È di fatto analogo al sistema RTK, con la differenza che non richiede l'uso di equipaggiamenti aggiuntivi. Spin3 GNSS è infatti un servizio pubblico gratuito che si appoggia ad un'infrastruttura interregionale costituita, ad oggi, da 39 stazioni permanenti. Nasce dall'unione delle precedenti reti regionali di Piemonte, Lombardia e Valle d'Aosta con lo scopo di

fornire un supporto migliorativo per applicazioni quali, ad esempio, misure topografiche, misurazioni di spostamento per edifici o versanti e, naturalmente, migliorare la precisione di collocamento di oggetti nello spazio, come nel caso in oggetto.

Come anticipato, l'infrastruttura è costituita da 39 antenne distribuite uniformemente sul territorio, si veda la figura 30, che, similmente ad un qualsiasi ricevitore GNSS, raccolgono i segnali inviati dai satelliti delle costellazioni GPS, GLONASS e Galileo.



**Figura 30:** Stazioni della rete Spin3

Le stazioni inviano poi le informazioni raccolte ad un centro di calcolo, questo ha il compito di rielaborare i dati grezzi con un triplice scopo. Testualmente:

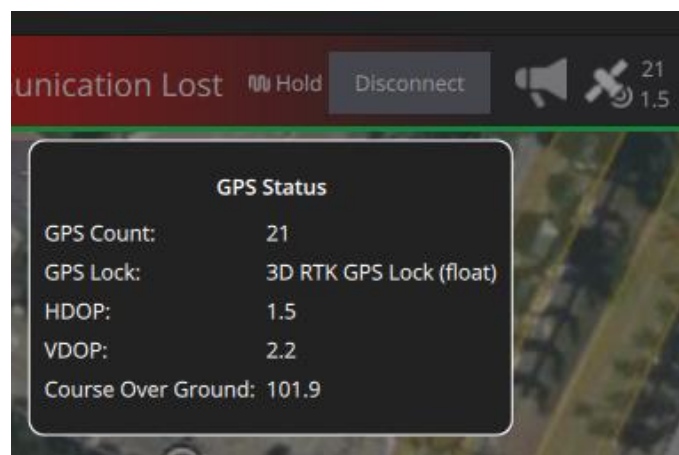
- stimare gli errori e i disturbi sulle singole stazioni;
- modellare gli errori e i disturbi nel tempo e nello spazio;
- distribuire all'utente in tempo reale i dati e i modelli, in modo che questa possa correggere i propri dati.

Nel caso in esame, inviando al drone i dati di correzione prodotti dal servizio Spin3 GNSS, è possibile raggiungere una precisione per il calcolo della posizione simile a quanto ottenibile con l'antenna RTK. Di fatto però, lo scopo della campagna di test effettuata nell'ambito di questa tesi ha proprio lo scopo di verificare questa possibilità, per valutare il livello di precisione ottenibile con i due metodi e consentire quindi un confronto diretto.



Dal punto di vista implementativo, è necessario aggiungere al workspace ROS sul companion computer un apposito pacchetto. Questo deve contenere dei nodi che, ricevuti i messaggi Spin3, li traducano ed inviino all'autopilota sul topic  $\mu$ ORB `gps_inject_data`; in questo modo i dati di correzione possono essere utilizzati dal veicolo per migliorare la precisione della posizione.

L'utente riceve conferma del buon funzionamento del sistema tramite QGroundControl: come mostrato in figura 31, visualizzando le informazioni relative al GPS, è possibile notare la transizione del GPS lock ad un grado superiore qualche istante dopo l'invio delle prime correzioni.



**Figura 31:** GPS lock RTK con Spin3, visualizzazione dei dati da QGroundControl

È bene precisare che, se la frequenza delle correzioni è insufficiente o se la qualità del segnale GNSS è scadente, potrebbe essere impossibile raggiungere i gradi RTK di GPS fix nonostante il buon funzionamento del sistema Spin3, dei nodi ROS e del middleware. Infine, per la ricezione dei messaggi di correzione dal server Spin3, è necessario che il companion computer sia connesso alla rete internet.

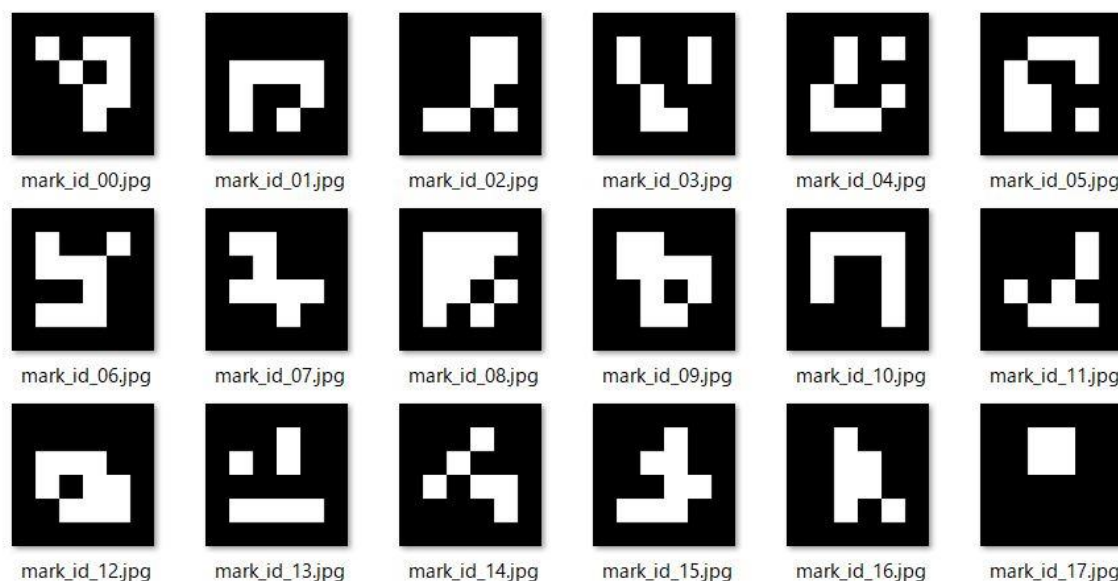
### 3.2.5 ArUco marker<sup>[46]</sup>

Per poter effettuare un'adeguata valutazione delle prestazioni e del diverso grado di precisione ottenuto sfruttando le diverse tecnologie per la determinazione della posizione in esame, risulta molto utile potersi affidare ad un riferimento esterno, fisso e con una posizione nota. Una possibile soluzione in tal senso è rappresentata dai marker ArUco, comunemente utilizzati per svariate applicazioni, dalla realtà aumentata agli atterraggi di precisione. In pratica però, il gruppo di ricerca non ha precedenti esperienze con l'utilizzo di

marker ArUco come riferimenti ground-truth e dunque uno degli obiettivi dei test effettuati nell'ambito di questa tesi riguarda lo stesso uso dei marker.

I marker di tipologia ArUco non sono gli unici disponibili, molto diffusi sono infatti anche quelli di tipo AprilTag, tuttavia il metodo di riconoscimento dei marker ArUco offre, in assoluto, la maggiore semplicità di implementazione con ROS.

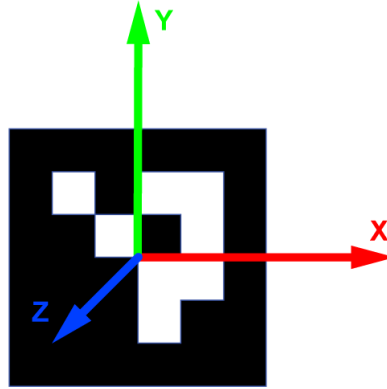
I marker ArUco, similmente ad altri tipi di marker, sono di fatto dei disegni di forma quadrata composti da pixel bianchi e neri che risultano quindi facili da identificare da parte di un calcolatore. Per semplicità di comprensione si fornisce una figura con qualche esempio.



**Figura 32:** Esempi di ArUco marker

Com'è possibile notare, i disegni sono alquanto diversi tra loro, in modo tale che i marker siano facilmente distinguibili gli uni dagli altri qualora ne siano presenti più d'uno all'interno dell'ambiente di missione. Anche l'asimmetria dei disegni non è casuale, l'osservazione dei marker ha lo scopo di ottenere informazioni di posizione relativa tra la camera ed il marker sia in termini lineari che rotativi: perché ciò sia possibile è necessario che la camera possa "vedere" un sistema di riferimento del marker per valutarne la distanza e l'orientamento, per poter determinare l'orientamento di tale sistema di riferimento è fondamentale che il disegno sia asimmetrico.





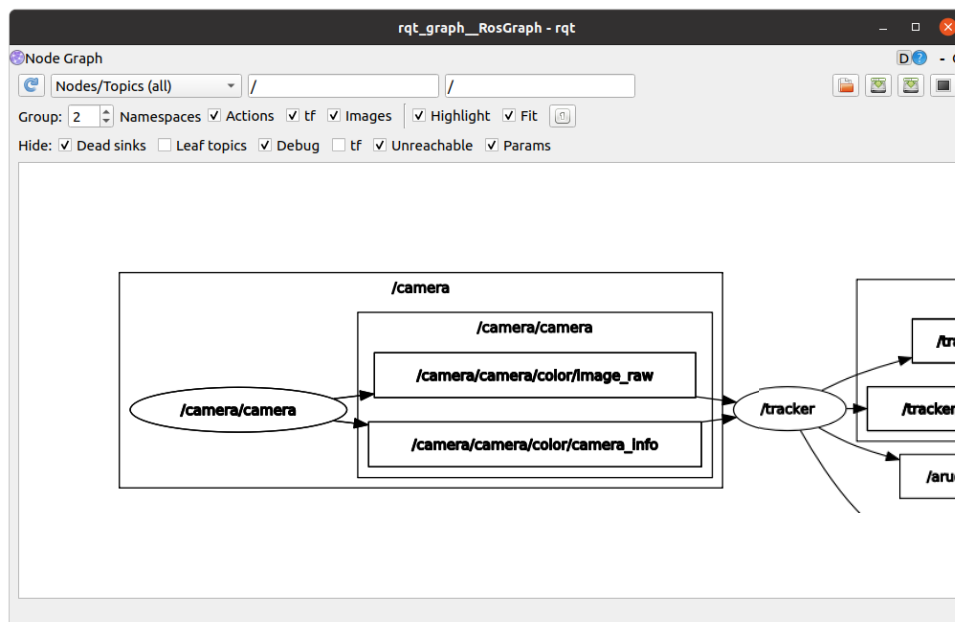
**Figura 33:** ArUco marker (id 00) con sistema di riferimento

Il primo passo per poter sfruttare questo strumento consiste nel collocare, nella zona in cui il veicolo opera, uno o più marker a seconda delle esigenze. Per quanto riguarda il veicolo in sé, è necessario che questo sia dotato di una fotocamera e di un calcolatore che elabori le immagini ricevute mediante un algoritmo di tracciamento. L'utente è poi tenuto a fornire all'algoritmo la misura della dimensione del marker, dato fondamentale per poter correttamente computare le distanze. A valle dell'elaborazione, l'algoritmo fornisce in output otto valori numerici che rappresentano l'id del marker individuato, la sua posizione lineare in tre coordinate e la sua orientazione espressa mediante i quaternioni.

Considerando lo specifico caso di studio in esame, è sufficiente utilizzare un marker posto al di sotto dell'area di volo del drone. In questo modo, grazie al tracking del marker, il veicolo manterrà un costante ed assoluto riferimento di posizione. In questo modo sarà quindi possibile valutare le prestazioni dei dispositivi di posizionamento satellitari ma anche la validità del marker come riferimento.

Per quanto riguarda la dotazione hardware, è già stata sottolineata la necessità di equipaggiare il drone con una camera rivolta verso il basso che sia così capace di mantenere costantemente il marker all'interno dell'inquadratura. Naturalmente ciò non è sufficiente, è infatti necessario provvedere anche all'adeguamento della configurazione software tramite l'installazione dell'algoritmo di tracking del marker. Per quanto riguarda l'algoritmo, sono disponibili per l'installazione alcune implementazioni mediante nodi ROS che offrono la possibilità di un'agile integrazione all'interno del framework già configurato. Il requisito da rispettare è naturalmente la compatibilità con la versione di ROS 2 Foxy, dopodiché sarà possibile, a valle dell'installazione, effettuare la compilazione del nuovo pacchetto e verificarne la funzionalità. Si

evidenzia ancora una volta la praticità che ROS offre, notando che il nodo di tracking ha il ruolo di subscriber nei confronti del topic su cui sono pubblicati, da un altro nodo ROS, i dati provenienti dalla fotocamera e che questo, a valle dell'elaborazione, pubblica a sua volta su un diverso topic i dati relativi all'ID e posizione del marker rilevato.



**Figura 34:** Schema dei nodi e topic ROS per camera e ArUco detection

Nello specifico il nodo `/camera/camera` è il responsabile dell'interfaccia con la fotocamera e ne pubblica i dati su svariati topic. Tra questi, il topic `.../image_raw` è quello da cui il nodo `/tracker` preleva i dati delle immagini raccolte dalla camera RGB. Quest'ultimo nodo è il responsabile del processamento delle immagini per l'individuazione della posizione del marker: se nell'immagine è presente un marker ArUco, allora il nodo `/tracker` ne pubblicherà, sul topic `/aruco_detections`, le informazioni di id e posizione. È fornito in figura 35 un riscontro di quanto descritto: grazie al comando `topic echo` di ROS è possibile visualizzare sul terminale i messaggi pubblicati sul topic `/aruco_detections` per avere un riscontro del buon funzionamento del sistema di identificazione del marker e della sua posizione.

```
pixhawkvision@pixhawkvision-UP-CHCR1: ~/camera_launch_ws
pixhawkvision@pixhawkvision-UP-CHCR1:~/camera_launch_ws$ ros2 topic echo /aruco_
detections
header:
  stamp:
    sec: 1731575105
    nanosec: 822632813
  frame_id: camera_color_optical_frame
markers:
- marker_id: 0
  pose:
    position:
      x: -0.007673422069039907
      y: 0.09905246449649283
      z: 0.4974996520587109
    orientation:
      x: 0.028069051062426388
      y: 0.9936476105003632
      z: -0.09655663615803696
      w: -0.05053088692292893
boards: []
---
header:
  stamp:
    sec: 1731575105
```

**Figura 35:** Visualizzazione esemplificativa dei dati pubblicati sul topic /aruco\_detections

### 3.2.6 Setup sperimentale

Nel presente capitolo si è spiegato come il drone calcola la propria posizione, quali modifiche hardware è stato necessario apportare per la campagna di test ed infine quali tecnologie per la posizione verranno testate. Prima di poter mostrare e discutere i risultati ottenuti nel capitolo a seguire, è necessaria un'ultima integrazione per descrivere il setup software e l'allestimento necessario per poter effettuare i test veri e propri.

In primis si precisa che le prove eseguite avranno come primo scopo quello di validare l'intervento di configurazione dimostrandone il buon funzionamento in volo. Segue una campagna costituita da tre test: uno in cui testare le funzionalità del solo ricevitore GNSS a, uno in cui verrà utilizzata la base RTK ed infine uno in cui verranno impiegate le correzioni Spin3 GNSS. Tutte le prove consisteranno in un volo quasi a punto fisso al di sopra del marker ArUco permettendo un confronto con il sistema di tracciamento del marker.

La raccolta dei dati per le analisi in post processing avverrà grazie allo strumento ROS bag di cui si è parlato nel paragrafo 2.3.1: grazie al middleware  $\mu$ XRCE-DDS infatti, anche i messaggi scambiati sui topic  $\mu$ ORB all'interno dell'autopilota possono essere ripubblicati su topic ROS, rendendoli quindi registrabili con ROS bag su un unico file unitamente con i dati prodotti dai nodi ROS come, ad esempio, quello responsabile per il tracciamento del marker. I

dati registrati saranno quindi quelli relativi a: posizione locale e globale del veicolo, dati GPS non processati, dati di tracciamento del marker e le letture del distance sensor per un riferimento verticale preciso.

Durante lo svolgimento dei test, per controllare il veicolo e tutte le relative funzionalità implementate, è naturalmente necessario disporre di un radiocomando ed un computer desktop (con il ruolo di ground station) per interagire con l'autopilota e monitorare in tempo reale il volo del drone. Oltre a questo, per la gestione dei nodi ROS e dei bag file, è necessario un secondo computer desktop connesso al companion computer mediante il protocollo SSH; sarà in questo modo possibile accedere al terminale del companion computer per lanciare i processi, quali nodi ROS e bag file ad esempio, che si necessita eseguire durante i voli di test.

Sfortunatamente, per l'avvio di ciascun processo è necessario lanciare una finestra del terminale esclusivamente dedicata. A tal proposito, per sveltire la procedura di avvio di ciascun test e diminuire il numero di finestre del terminale da aprire in SSH, si è deciso di ricorrere all'utilizzo di shell script e launch file. Uno shell script, lanciabile semplicemente indicandone il percorso ed il nome, ha lo scopo di raccogliere una serie di istruzioni che dovrebbero altrimenti essere inserite manualmente una ad una sul terminale. Il primo shell script creato ha quindi lo scopo di aprire e configurare il workspace di ROS e quindi di richiamare un launch file; il launch file a sua volta lancia i nodi ROS della fotocamera RealSense e del tracciamento del marker oltre al bag file di cui al capoverso precedente. È comunque necessaria una seconda finestra SSH per poter lanciare il middleware  $\mu$ XRCE-DDS. Per effettuare il test con le correzioni Spin3 GNSS è infine richiesta una terza finestra per lanciare un altro shell script; questo è simile al precedente, ma richiama un diverso launch file, quello che avvia i nodi per la raccolta, traduzione ed invio all'autopilota dei messaggi di correzione della posizione.

## 4) Risultati

Lo scopo di questo capitolo è quello di presentare i risultati dei test effettuati nell'ambito della campagna sperimentale. Nella sezione 4.1 si descrive il volo di prova condotto per validare la configurazione del veicolo, mentre nella sezione 4.2 vengono raggruppati i test dei sistemi di determinazione della posizione.

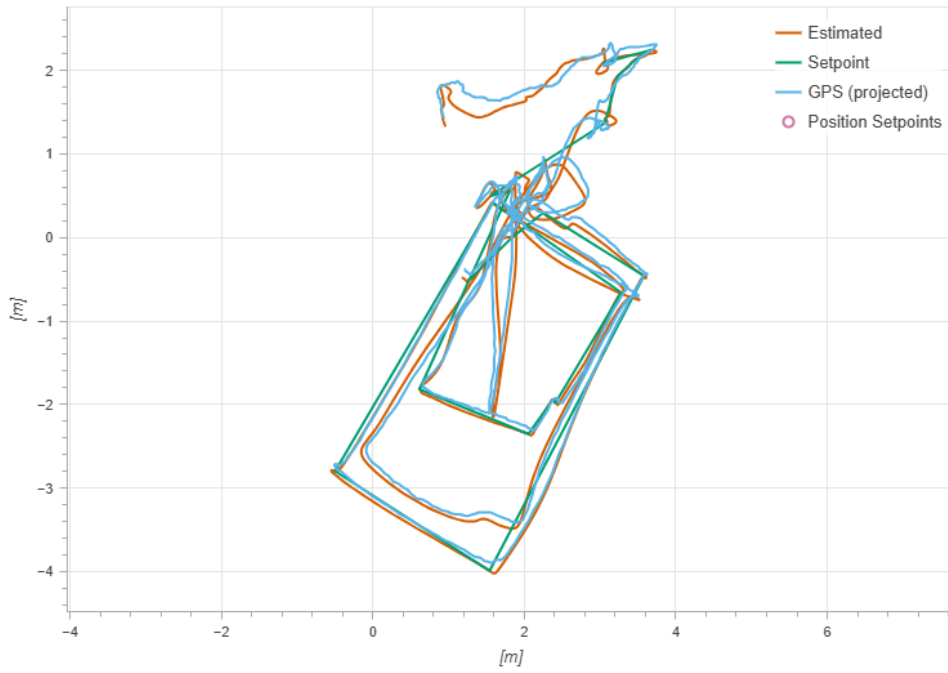
### 4.1 Validazione della configurazione

A monte dello svolgimento di questa attività di tesi il drone PX4 Vision Autonomy Development Kit non rispondeva nel miglior modo possibile alle esigenze del gruppo di ricerca: è stato quindi condotto un intervento di aggiornamento e configurazione che ha coinvolto, come spiegato nel capitolo 2, il firmware dell'autopilota, il companion computer ed infine anche il protocollo di comunicazione tra questi. Gli interventi condotti sul software del veicolo ne hanno quindi coinvolto tutti gli elementi principali, rendendo quindi necessario osservare il comportamento del quadricottero in volo per confermarne il buon funzionamento o, in caso contrario, per procedere con eventuali ulteriori modifiche correttive.

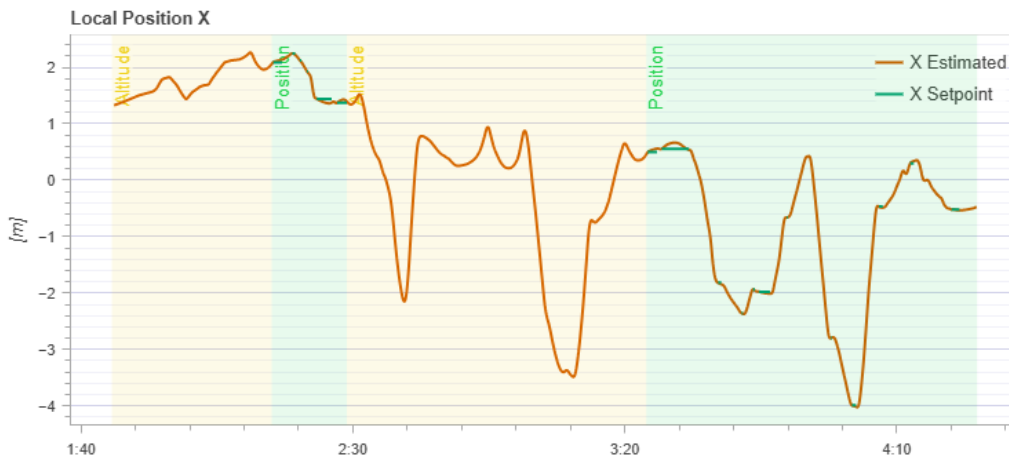
Sono quindi di seguito presentati i dati registrati dall'autopilota durante un volo di prova: è possibile notare come il veicolo si sia dimostrato pienamente funzionante.

Com'è possibile osservare dai grafici nelle figure a seguire, dopo la manovra di decollo sono stati impartiti al veicolo comandi di movimento in tutte e tre le direzioni e di imbardata per terminare con l'atterraggio.

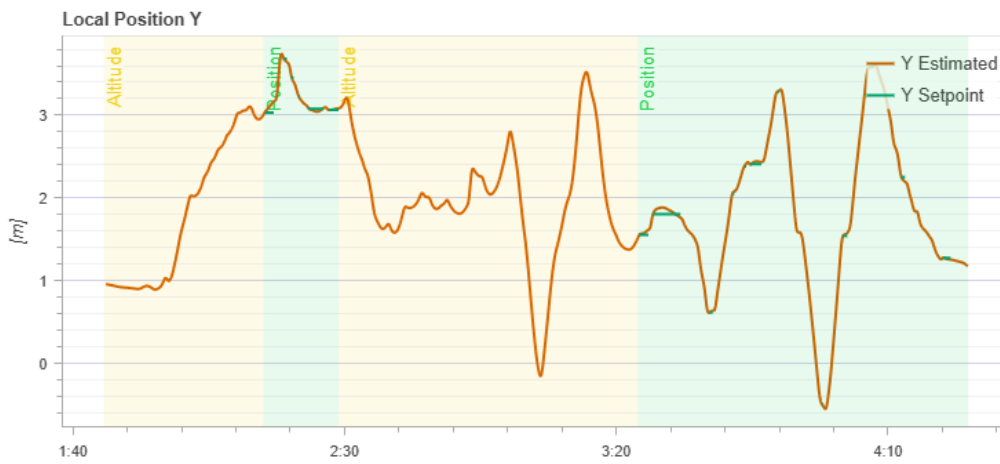
Nel dettaglio, la figura 36 mostra i movimenti del drone sul piano x-y, sia secondo i dati GPS che secondo la stima effettuata dall'autopilota grazie ai sensori di bordo. Si notino i movimenti lungo le direzioni laterale e longitudinale ed anche la buona sovrapposibilità della posizione del veicolo con il setpoint impartito. A completamento, le figure a seguire (37 e 38) mostrano il movimento nel tempo rispetto alle due direzioni x e y.



**Figura 36:** Spostamenti nel piano x-y

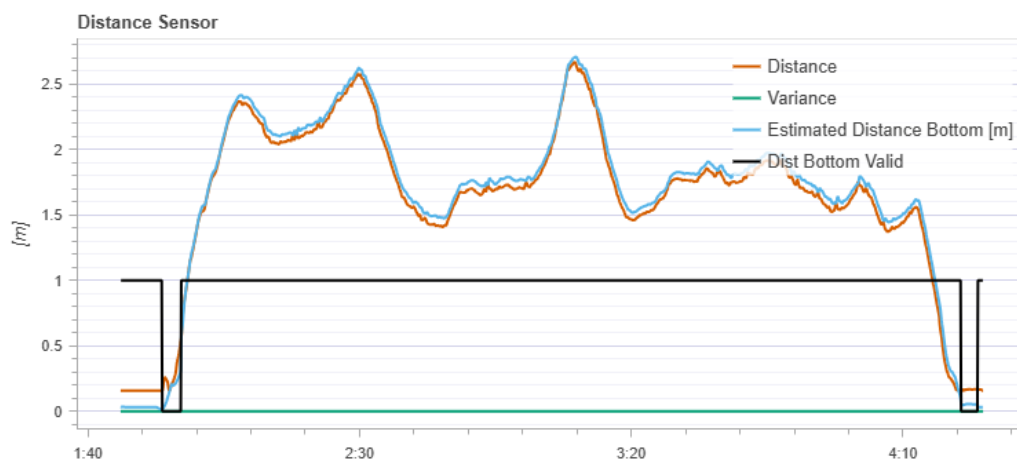


**Figura 37:** Spostamento longitudinale nel tempo



**Figura 38:** Spostamento laterale nel tempo

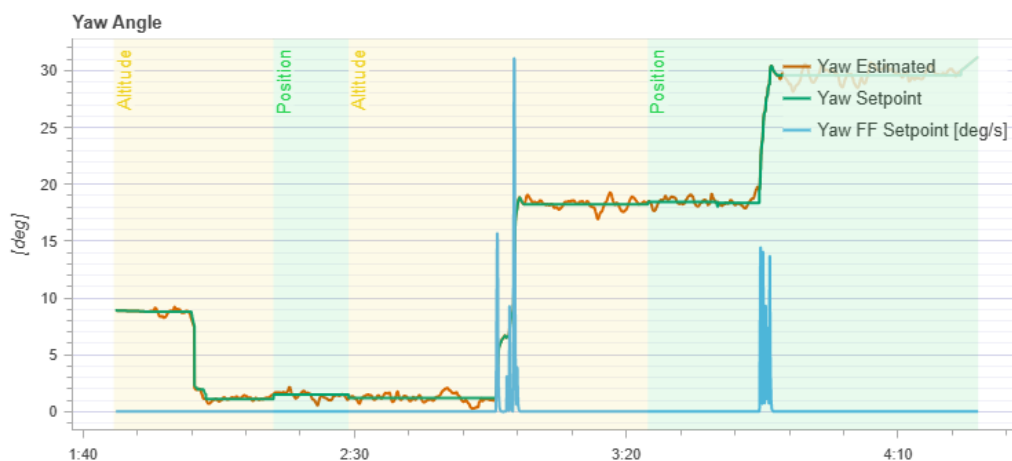
La figura 39 mostra poi i movimenti verticali come misurati dal distance sensor dell'optical flow.



**Figura 39:** Spostamento verticale secondo le misurazioni del distance sensor

In sintesi, a seguito della manovra di decollo sono stati effettuati dei cauti movimenti verticali senza eccessivi spostamenti nel piano, per poi procedere, nella seconda parte del volo, con più ampi movimenti orizzontali ad una quota circa costante.

La figura 40 mostra poi le variazioni dell'angolo di yaw: anche in tal senso sono stati impartiti dei comandi come ben visibile dalle variazioni del setpoint.

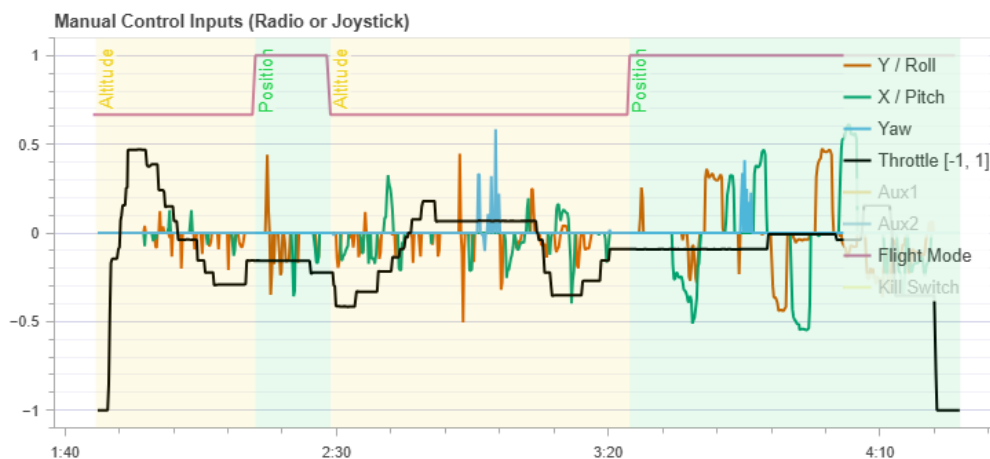


**Figura 40:** Valori assunti dall'angolo di yaw durante il volo

Infine, la figura 41 mostra per completezza i comandi impartiti al veicolo. Quel che si intende far notare in particolare modo è il cambio di modalità dell'autopilota<sup>[49]</sup>: quando il veicolo si trova in modalità altitude la leva di throttle del radiocomando controlla il rateo di salita, mentre la leva di pitch e roll controlla gli spostamenti sul piano orizzontale; con le leve in posizione neutra il quadricottero si mantiene ad un'altezza costante mentre è soggetto a drift nel



piano. La modalità position è in parte simmetrica: in questa modalità anche la leva di pitch e roll comanda delle accelerazioni ed il veicolo mantiene autonomamente la posizione nel piano quando la leva è in posizione neutra; in questo caso però è la quota a non essere bloccata sicché gli spostamenti verticali sono affidati al controllo manuale da parte dell'operatore. Uno degli scopi di questo primo volo di prova è anche quello di verificare che il veicolo funzioni correttamente in entrambe le modalità e che non manifesti comportamenti anomali in caso di cambio di modalità durante il volo.

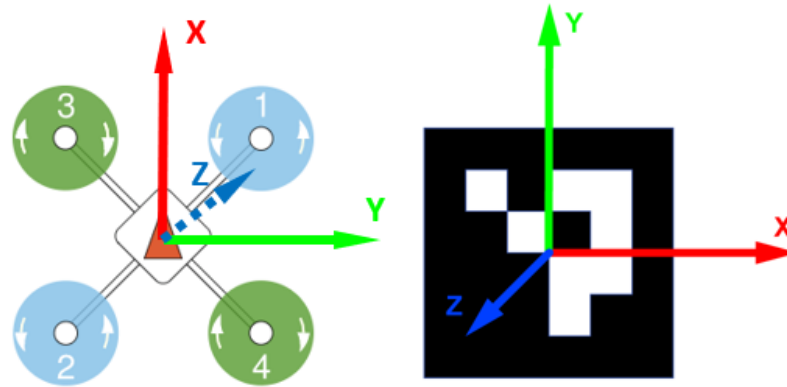


**Figura 41:** Comandi impartiti durante il volo

## 4.2 Test dei sistemi per la determinazione della posizione

In questa sezione verranno presentati i risultati dei tre seguenti test: il primo prevede il confronto della posizione calcolata dal veicolo con il riferimento dato dal marker ArUco, il secondo prevede l'utilizzo della base RTK ed infine il terzo vedrà la base RTK sostituita dalle correzioni Spin3 GNSS. L'obiettivo è quindi quello di verificare le prestazioni dei diversi sistemi e la precisione raggiungibile in ogni caso, oltre alla validità del tracciamento del marker come riferimento.

Per permettere un confronto tra i diversi dati raccolti è stato necessario effettuare alcune conversioni per arrivare infine ad esprimere tutti gli spostamenti secondo gli assi corpo del quadricottero. I dati GPS non processati e quelli di posizione globale sono espressi in termini di latitudine e longitudine; questi sono stati convertiti in spostamenti relativi in metri rispetto alle coordinate del punto iniziale del volo. Anche le misure di tracciamento del marker hanno richiesto una conversione, perché, come visibile in figura 42, il sistema di riferimento del marker non coincide con gli assi corpo del veicolo.



**Figura 42:** Sistemi di riferimento a confronto

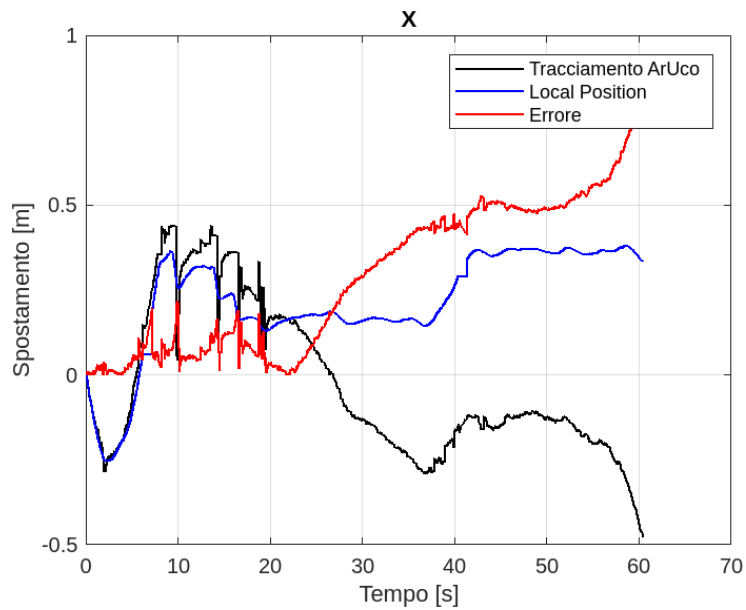
Si precisa che, per poter convertire i dati GPS secondo gli assi corpo del veicolo, è necessario conoscere l'orientamento di quest'ultimo rispetto al nord. Per semplicità si è deciso di disporre il marker ed il drone in modo tale che l'asse X body e Y del marker fossero paralleli e rivolti verso il nord. Si è poi evitato di impartire comandi di yaw per mantenere l'allineamento per l'intera durata dei test, garantendo così la semplicità delle conversioni.

#### 4.2.1 Modulo GNSS e ArUco marker

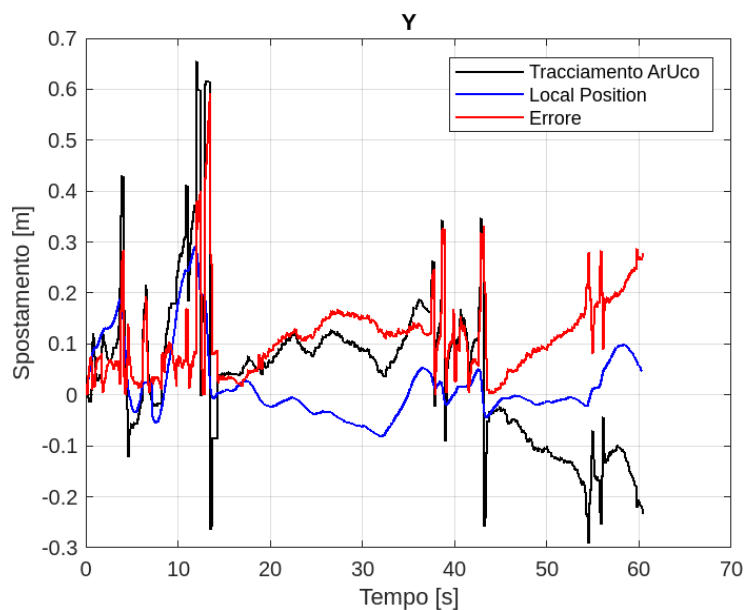
Con il primo test si intende mettere a confronto la capacità del veicolo di determinare la propria posizione a partire dai dati ottenuti grazie alle misure del modulo GNSS, con i dati di tracciamento del marker.

Per prima cosa vengono mostrati (nelle figure 43-44-45) i grafici che mostrano gli spostamenti lungo le tre direzioni durante il volo, permettendo un confronto tra i dati di tracciamento del marker e la posizione locale calcolata dall'autopilota. È poi presente una curva di errore calcolata come differenza delle altre due.

Per quanto riguarda gli spostamenti in X e in Y, è possibile notare un andamento prevalentemente concorde delle due curve, sebbene con un errore mediamente vicino a 20-30 cm con picchi anche oltre il mezzo metro. Questi risultati sono in linea con le prestazioni che ci si può aspettare da un valido modulo GNSS come il F9P adoperato in questo caso. Si nota comunque una tendenza di crescita dell'errore con il passare del tempo, probabilmente dovuto ad un leggero drift del GPS.



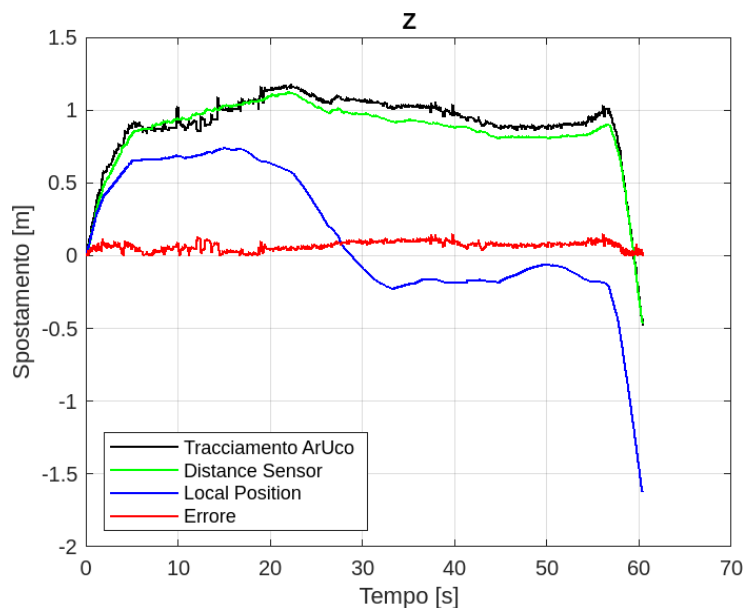
**Figura 43:** Volo con GPS – Spostamento longitudinale



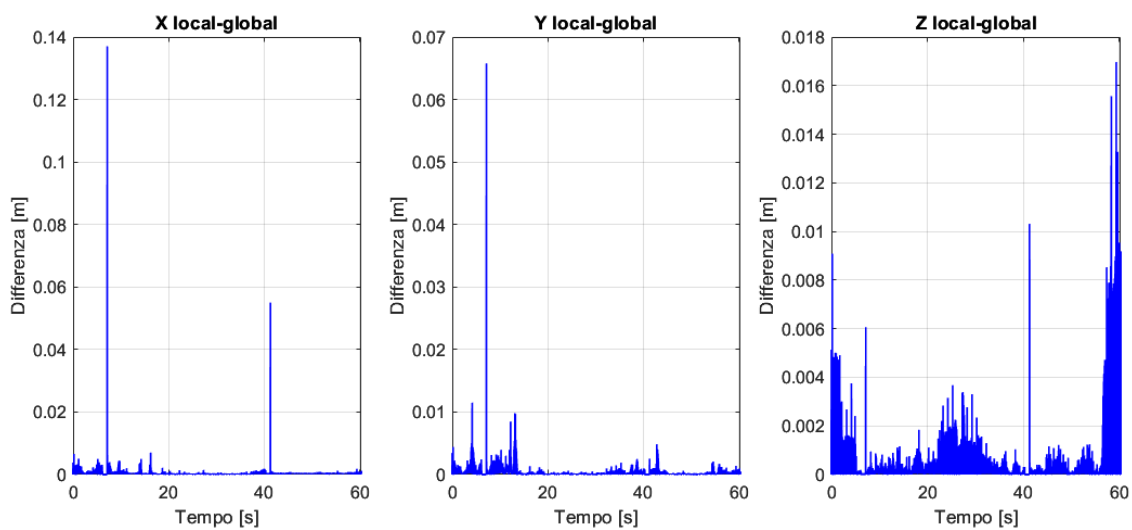
**Figura 44:** Volo con GPS – Spostamento laterale

Per quanto riguarda lo spostamento in Z (figura 45) sono necessarie alcune precisazioni: essendo il veicolo dotato di un distance sensor, si è deciso di sfruttarne le letture come ulteriore riferimento; questo grafico presenta dunque una curva aggiuntiva e l'errore è, in questo caso, calcolato tra le misure del distance sensor e di posizione del marker. Questa scelta permette di verificare l'efficacia del marker quale riferimento: l'errore rimane costantemente entro i 10 cm ed è principalmente causato dal rumore dei dati del marker. A proposito di questo fenomeno, la causa potrebbe stare nel fatto che il veicolo trasmette

vibrazioni alla camera usata per il tracciamento del marker, i dati non vengono filtrati ed infine hanno una frequenza di campionamento inferiore rispetto ai dati registrati invece dai topic  $\mu$ ORB dell'autopilota. Anche in questo caso si può notare, a partire dalla metà del volo circa, una crescente discrepanza tra i dati di tracciamento del marker e del GPS, ma grazie al riferimento del distance sensor, possiamo concludere che si tratta di un drift del sistema GNSS; è quindi possibile riportare questa conclusione anche allo spostamento lungo le altre due direzioni, rispetto alle quali si notava lo stesso fenomeno.



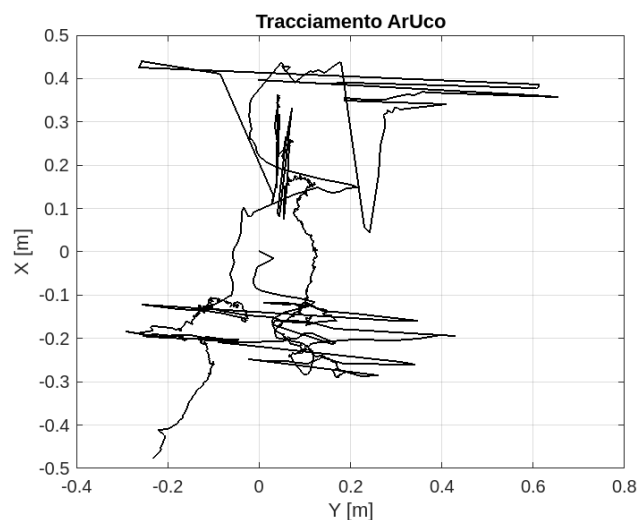
**Figura 45:** Volo con GPS – Spostamento verticale



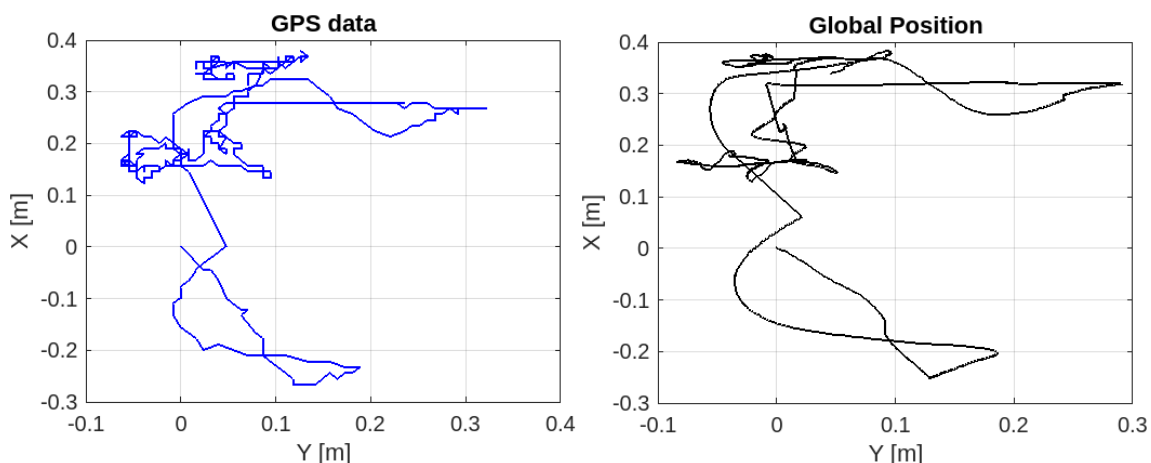
**Figura 46:** Confronto tra posizione locale e globale

Nelle figure 43, 44 e 45 non sono inclusi i dati di posizione globale, ma solo quelli di posizione locale; la giustificazione per questa scelta si trova in figura 46. Il grafico in essa riportato mostra la differenza tra posizione locale e posizione globale trasformata da coordinate geografiche a spostamenti relativi in assi corpo. Concordemente con quanto già affermato nel paragrafo 3.2.1, possiamo notare che queste due posizioni sono di fatto coincidenti, con un errore prevalentemente millimetrico e molto probabilmente dovuto alla sola approssimazione dei dati. Grazie a questa conclusione vengono totalmente omesse sovrapposizioni di entrambe le informazioni in un medesimo grafico.

Vengono quindi forniti tre grafici con i movimenti nel piano orizzontale, in figura 47 sono mostrati i dati di tracciamento del marker, in figura 48 la posizione globale ed i dati GPS non processati.

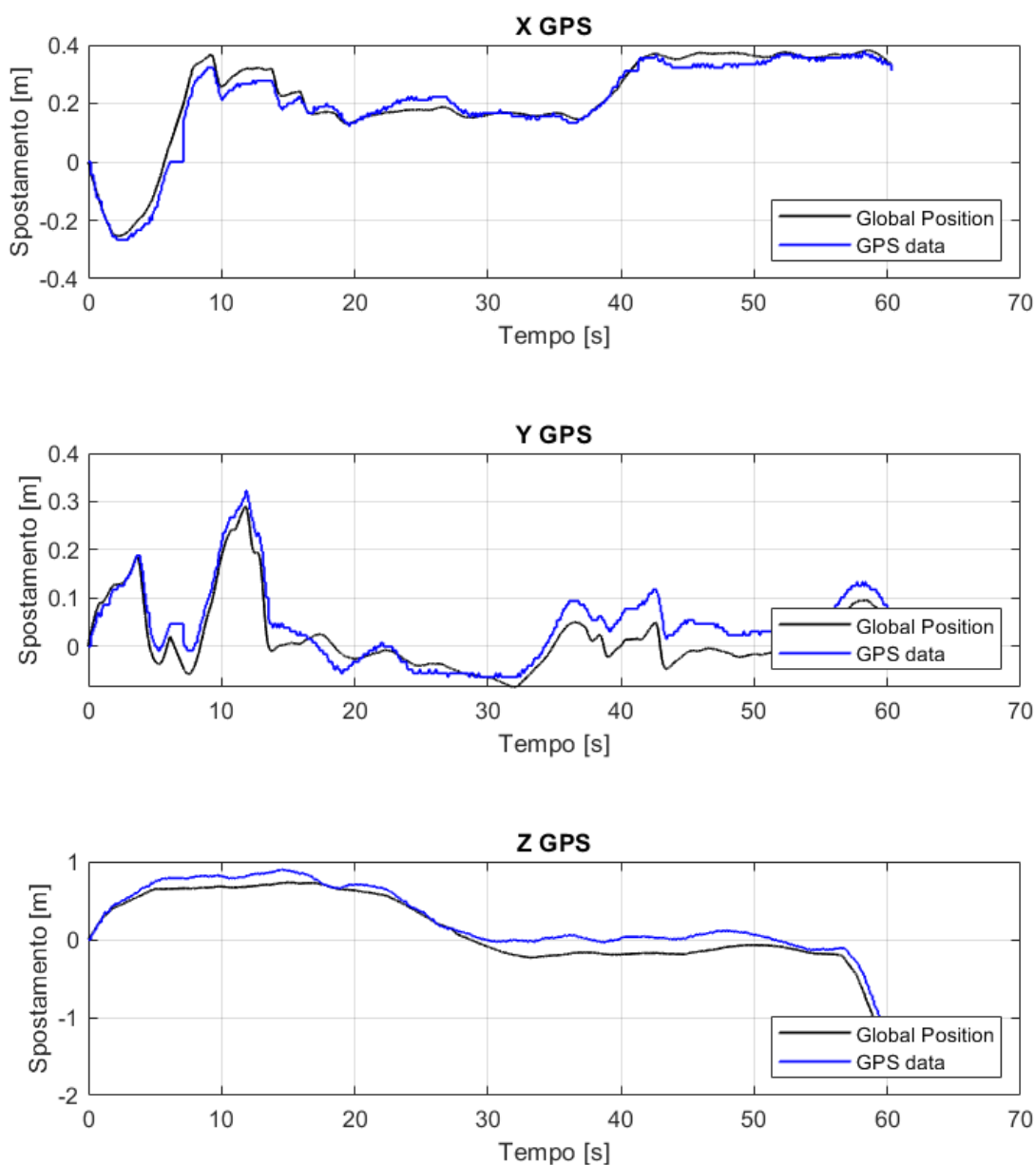


**Figura 47:** Volo con GPS – Spostamenti nel piano (marker)



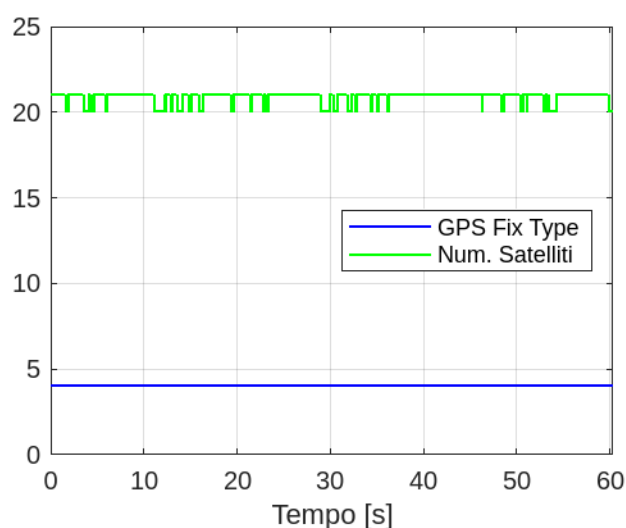
**Figura 48:** Volo con GPS – Spostamenti nel piano (dati GPS e global position)

Per prima cosa si noti come la figura 47 non sia pienamente concorde con i grafici in figura 48, ma è già stato stabilito che ciò dipende da un leggero drift del GPS. Inoltre, il grafico con i dati di tracciamento del marker mette particolarmente in evidenza gli effetti del rumore, che rende più complessa la lettura della traiettoria reale; non solo, le correzioni di posizione del veicolo dipendono da variazioni di assetto, quali ad esempio cambiamenti nell'angolo di rollio per la correzione della posizione laterale: i cambiamenti d'assetto portano però ad uno spostamento del marker nel campo visivo della camera di tracciamento, provocando i marcati "zig-zag" che non trovano riscontro nei dati di posizione.



**Figura 49:** Volo con GPS – effetto del sensor fusion

Dalla figura 48 è poi possibile riconoscere l'effetto del sensor fusion, ulteriormente evidenziato dai grafici in figura 49: nel paragrafo 3.2.1 si è parlato di come i dati GPS vengano filtrati e combinati con quelli degli altri sensori del veicolo per calcolarne la posizione locale. Si nota infatti dai grafici come i dati GPS siano in generale più imprecisi e rumorosi rispetto ai più raffinati dati di posizione calcolati dall'autopilota. Si veda a tale proposito anche la figura 50, che mostra il numero di satelliti visibili ma soprattutto il GPS fix type: si nota che il suo valore è 4 per tutta la durata del volo e si ricorda che tale valore corrisponde al fix type "3D DGPS Lock", ad indicare che le informazioni di posizione sono migliorate grazie al dead reckoning (si veda il paragrafo 3.2.2). È quindi normale osservare nelle figure 48 e 49 delle differenze tra i dati, a riprova dell'efficacia del filtro di Kalman esteso utilizzato dall'autopilota.



**Figura 50:** Volo con GPS – prestazioni del GPS

Infine, è riportato il numero di satelliti visibili perché determinante per le prestazioni del calcolo della posizione da parte del modulo GNSS. In questo caso, con oltre 20 satelliti in vista, è possibile ottenere delle buone prestazioni: è già stato fatto notare come l'errore mostrato nelle figure 43 e 44 sia sempre ampiamente inferiore al metro.

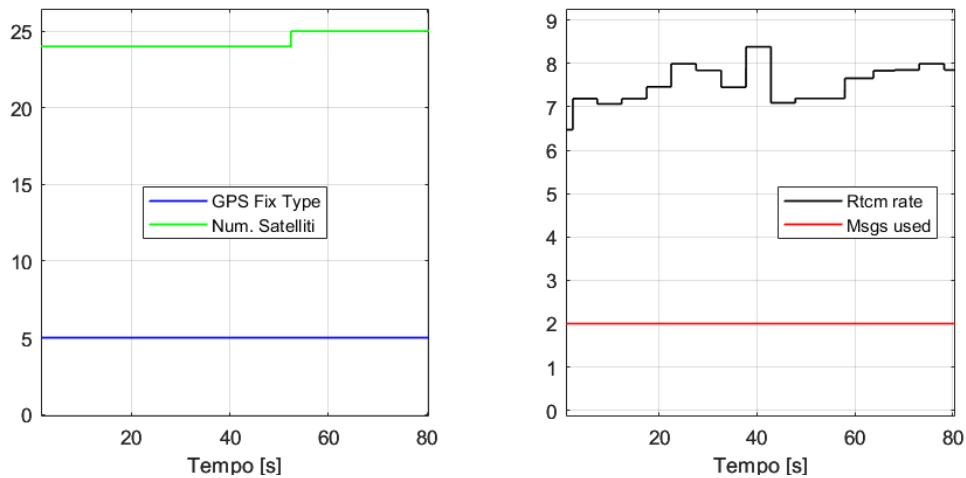
#### 4.2.2 Modulo GNSS, ArUco marker e antenna RTK

Per lo svolgimento del secondo test è stata aggiunta una base RTK come spiegato nel paragrafo 3.2.3. L'intento è quindi quello di verificarne l'efficacia e di riflesso l'impatto sulla precisione con cui il drone determina la propria posizione.



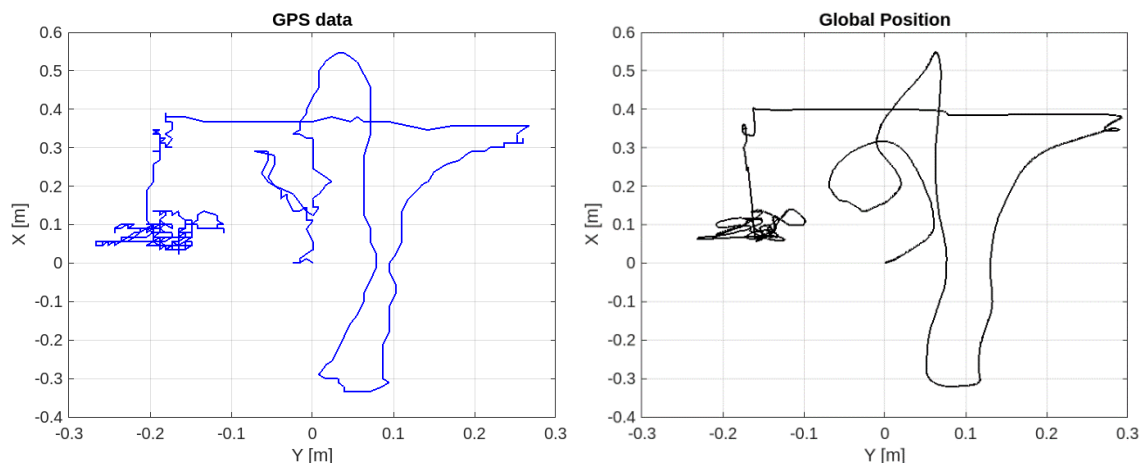
Per questo test, il primo grafico presentato (figura 51) è quello riferito alle prestazioni del gps.

### Prestazioni GPS



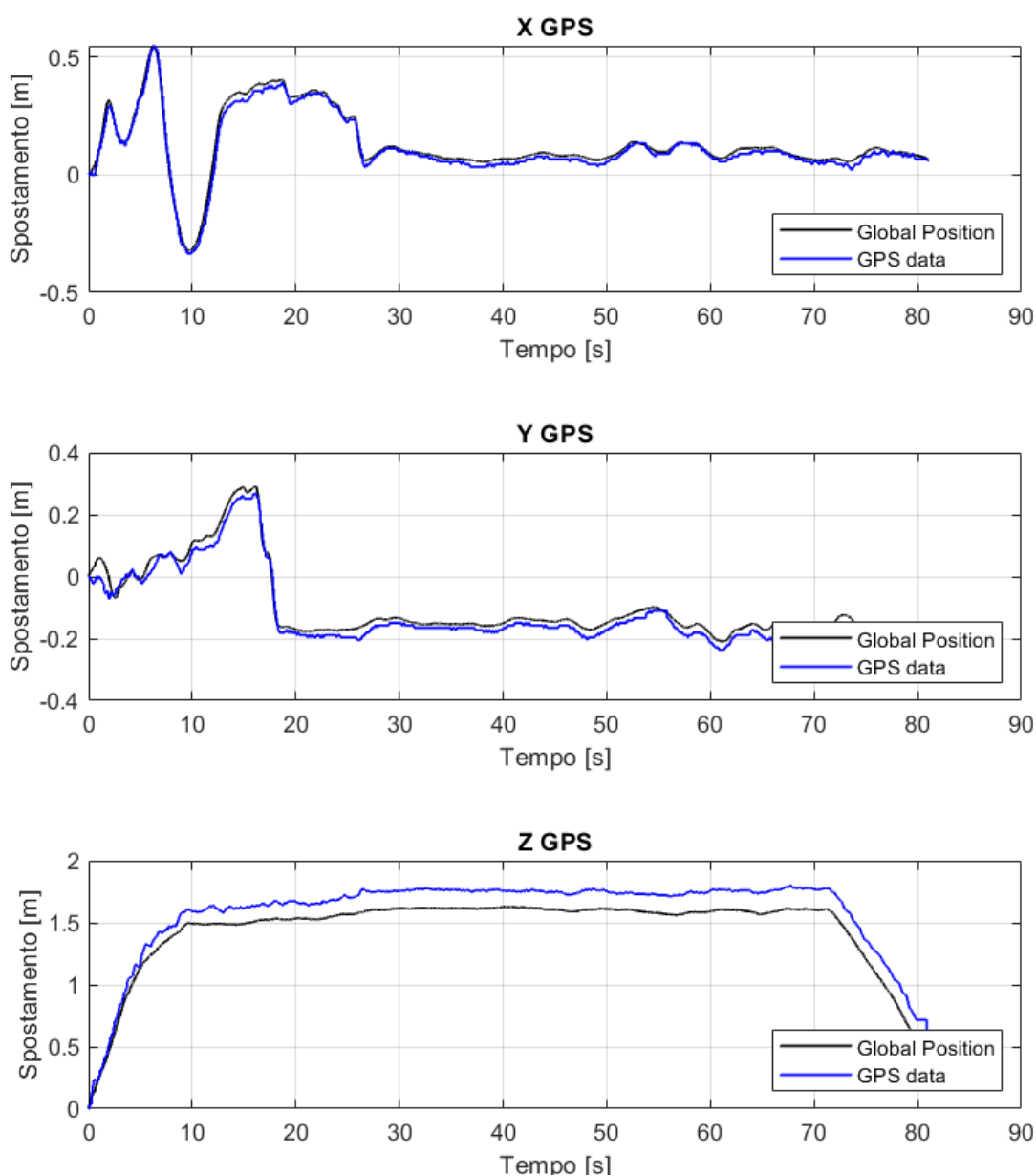
**Figura 51:** Volo con RTK – prestazioni del GPS

È possibile notare che il fix type in questo caso è 5, ovvero “3D RTK GPS Lock (float)”. È quindi naturale aspettarsi delle prestazioni migliori rispetto al test precedente, anche se non raggiungendo il livello 6 probabilmente non verrà raggiunta la precisione centimetrica. Il grafico di destra informa della frequenza con cui i messaggi di correzione vengono inviati (rtcm rate) e del fatto che tali messaggi vengano effettivamente accettati dall’autopilota (msgs used); questi numeri mostrano delle prestazioni accettabili, ma soprattutto confermano l’invio e, successivamente, la ricezione ed utilizzo dei messaggi da parte dell’autopilota. Il valore 2 per msgs used conferma che l’autopilota accetta i messaggi; se tale valore scendesse a 1 (messaggi non accettati) o a 0 (messaggi non ricevuti), ciò porterebbe ad un decadimento delle prestazioni e probabilmente anche del GPS fix al livello 4.



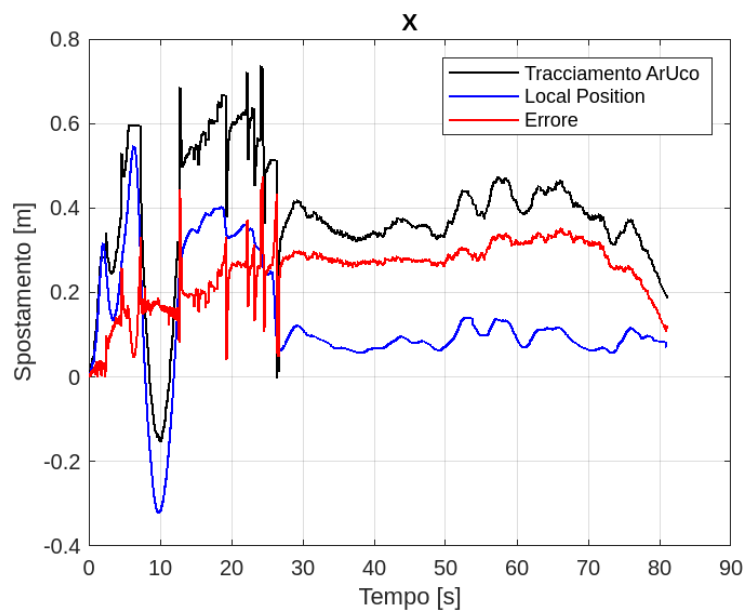
**Figura 52:** Volo con RTK – Spostamenti nel piano (dati GPS e global position)

La figura 52, se confrontata con l'analogica del volo precedente cioè la figura 48, permette di notare come in questo caso i dati GPS, sebbene comunque affetti da rumore, siano molto più prossimi a quelli relativi alla posizione globale. Anche la figura 53 permette di sottolineare lo stesso concetto: le curve mostrate dai grafici in essa contenuti mostrano come, grazie alla più elevata precisione dei dati GPS, l'effetto del sensor fusion generi una minore discrepanza tra i dati GPS non processati e la posizione globale. Di fatto la maggiore precisione dei dati di partenza richiede un minore intervento di correzione da parte dell'algorithmo dell'autopilota.

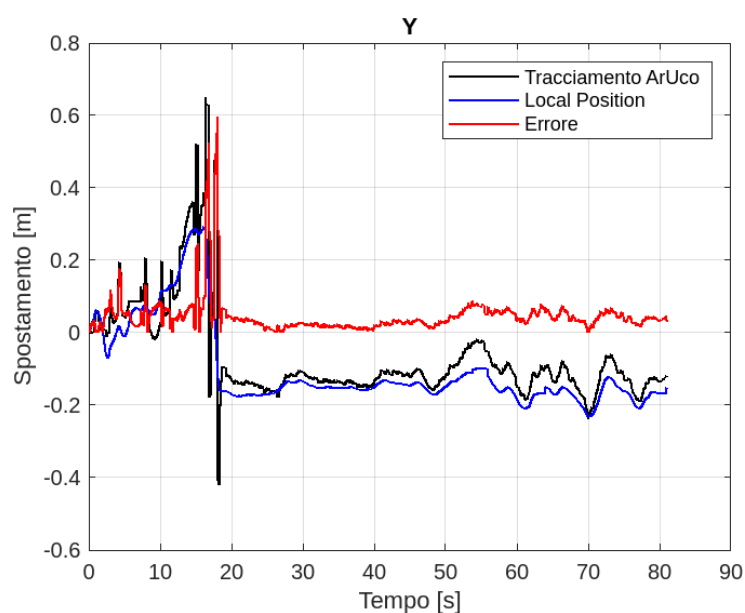


**Figura 53:** Volo con RTK – effetto del sensor fusion

Passando quindi al confronto con il tracciamento del marker, vengono forniti nelle figure 54, 55 e 56 i dati relativi agli spostamenti lungo i tre assi.



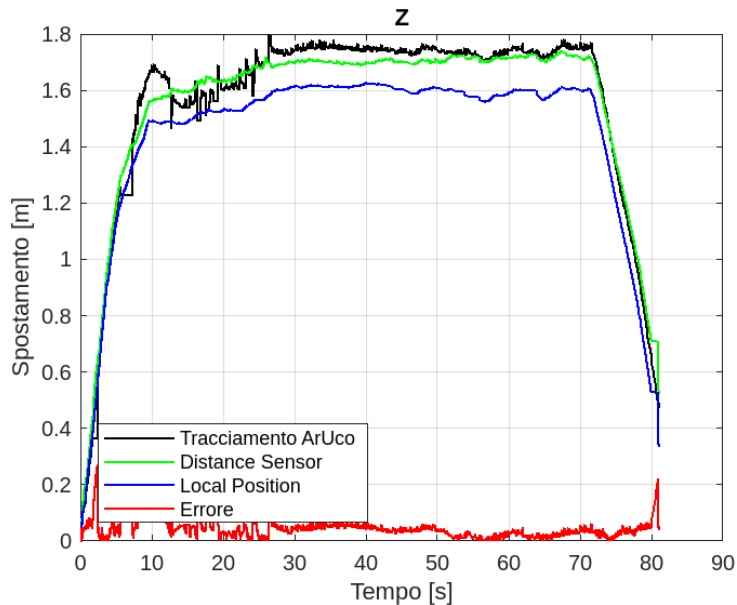
**Figura 54:** Volo con RTK – Spostamento longitudinale



**Figura 55:** Volo con RTK – Spostamento laterale

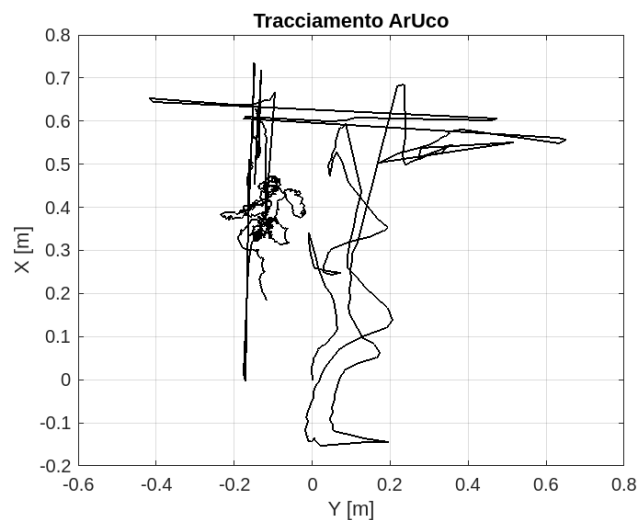
È possibile notare, soprattutto in direzione laterale un notevole aumento della precisione: nella fase iniziale del volo i dati di tracciamento manifestano dei picchi quasi certamente dovuti a variazioni di assetto, mentre al termine delle manovre iniziali, a partire da 20 s, il veicolo si stabilizza e i due metodi di misura concordano a meno di un errore quasi costante ed inferiore ai 10 cm. L'errore è

leggermente maggiore per la posizione lungo la direzione longitudinale, ma l'aspetto importante che emerge da tutti e tre i grafici è la totale assenza di fenomeni di drift. Anche rispetto alla direzione verticale, lungo la quale si mostravano forti discrepanze in figura 45, si nota un andamento concorde di tutti i tre sistemi di misura adoperati.



**Figura 56:** Volo con RTK – Spostamento verticale

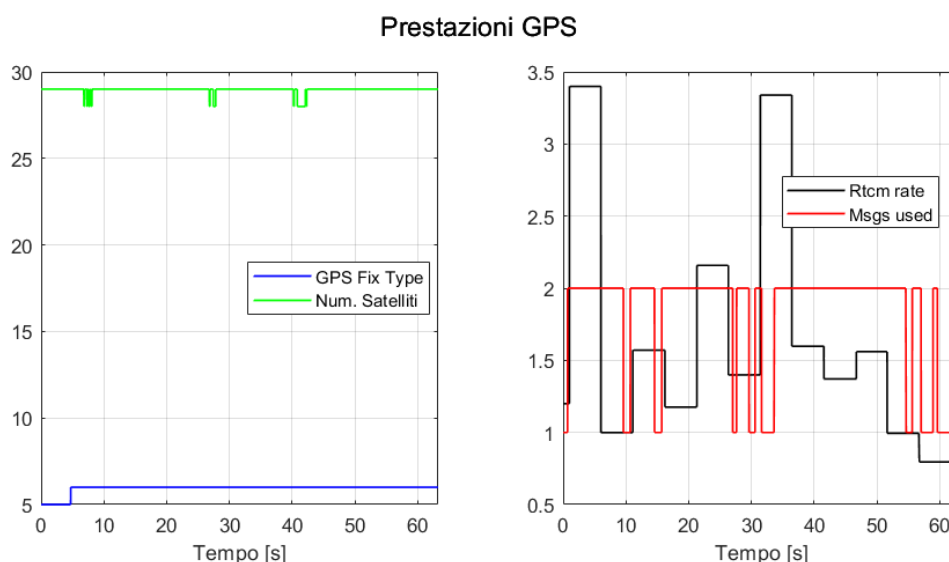
Infine, in figura 57 è mostrata la traiettoria nel piano orizzontale ricostruita con i dati di tracciamento del marker. Anche in questo caso si notano degli “zig-zag” dovuti all’assetto, ma per il resto i dati concordano con quelli di posizione globale e GPS (figura 52) più di quanto non avvenisse nel caso del test precedente.



**Figura 57:** Volo con RTK – Spostamenti nel piano (marker)

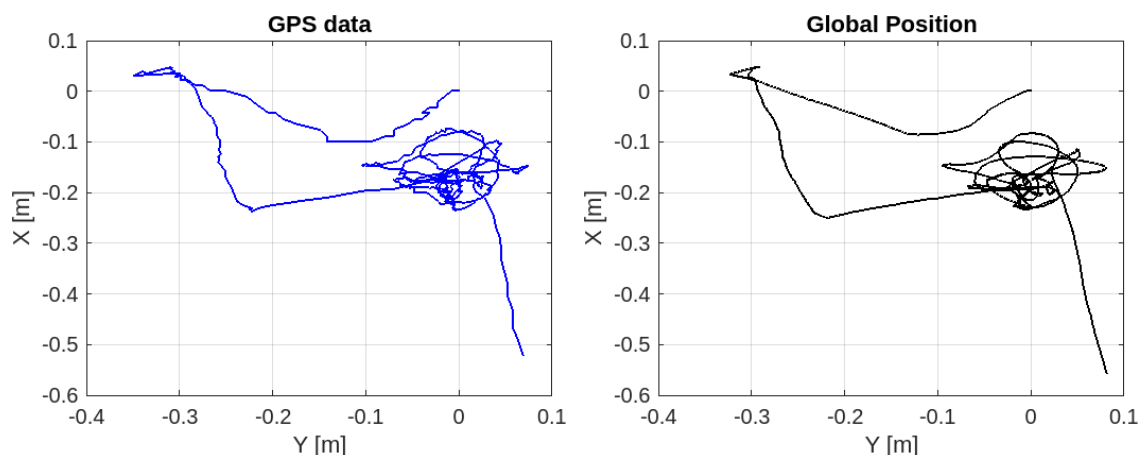
### 4.2.3 Modulo GNSS, ArUco marker e correzioni Spin3

Per lo svolgimento del terzo test la base RTK è stata sostituita dai messaggi di correzione Spin3 di cui si è parlato nel paragrafo 3.2.3, con l'intento di verificarne l'efficacia soprattutto rispetto all'utilizzo della stessa base RTK



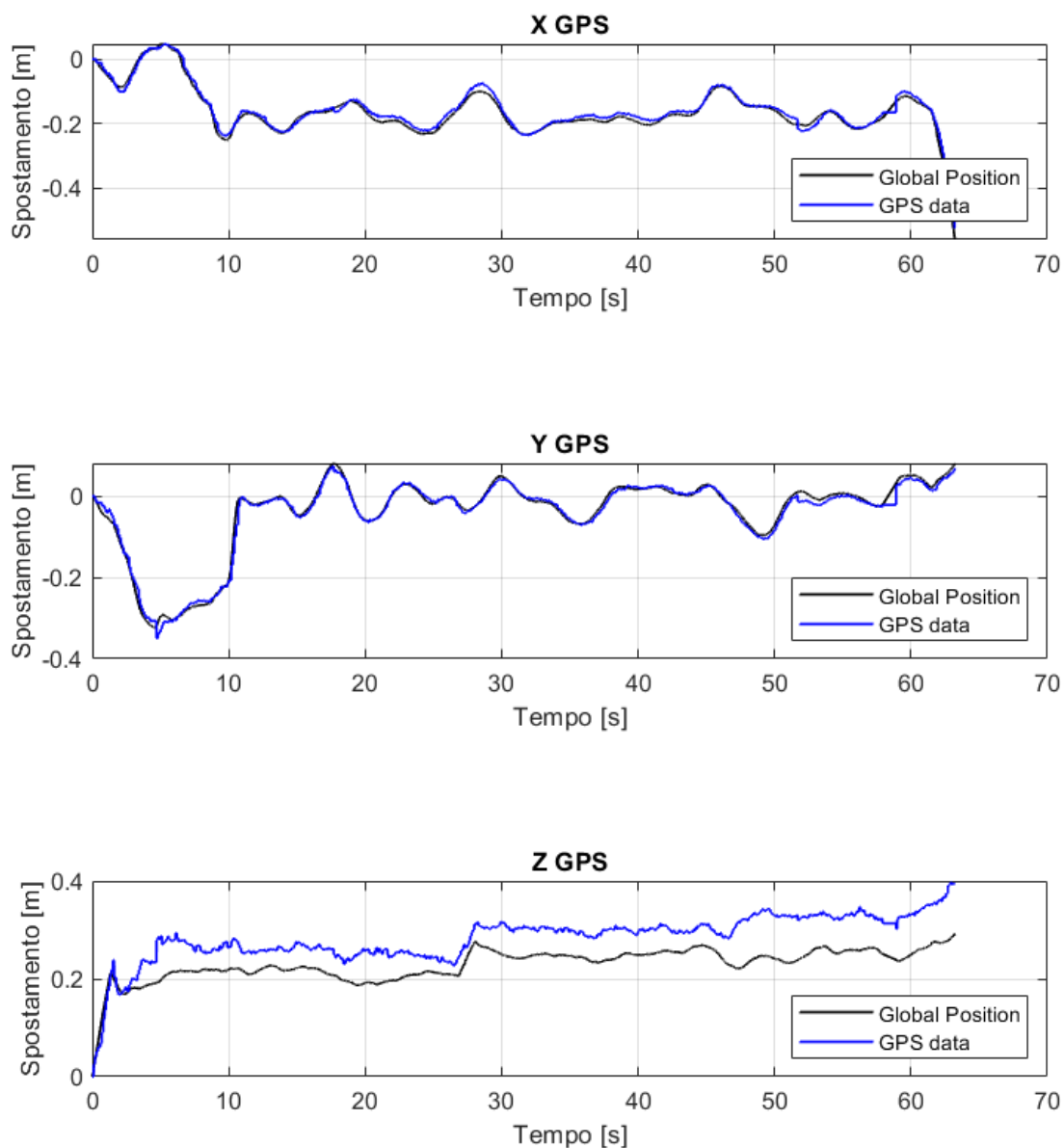
**Figura 58:** Volo con Spin3 – prestazioni del GPS

Dalla figura 58 è già possibile notare alcune differenze rispetto al test precedente: in questo caso il fix type è 6, ovvero "3D RTK GPS Lock (fixed)", da cui ci si aspetta di ottenere prestazioni ancor più positive, con il raggiungimento di precisioni centimetriche. È poi possibile notare che in questo caso la frequenza delle correzioni inviate è significativamente inferiore rispetto all'antenna RTK, tanto che di tanto in tanto, ed in particolare quando essa scende sotto il valore 1,5, anche il valore di msgs used passa da 2 a 1. Fortunatamente, come visibile nel grafico di sinistra, ciò non intacca il fix type e non dovrebbe quindi avere un impatto eccessivo sulle prestazioni.



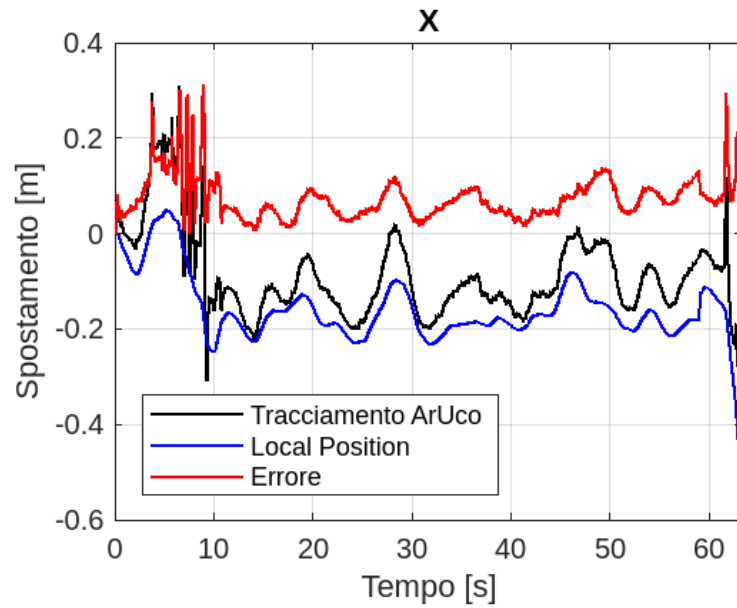
**Figura 59:** Volo con Spin3 – Spostamenti nel piano (dati GPS e global position)

Per quanto riguarda la figura 59, è evidente la somiglianza quasi totale delle traiettorie nel piano: la precisione dei dati GNSS in questo caso è talmente precisa da cogliere con precisione elevata i movimenti del veicolo. È possibile trarre la stessa conclusione anche dalla figura 60, dove si nota come la differenza tra i dati GPS e la posizione calcolata dall'autopilota coincidano a meno di un errore centimetrico. In direzione verticale la discrepanza è invece maggiore, sebbene sia comunque tra i 5 e i 10 cm, circa la metà rispetto al test con la base RTK.

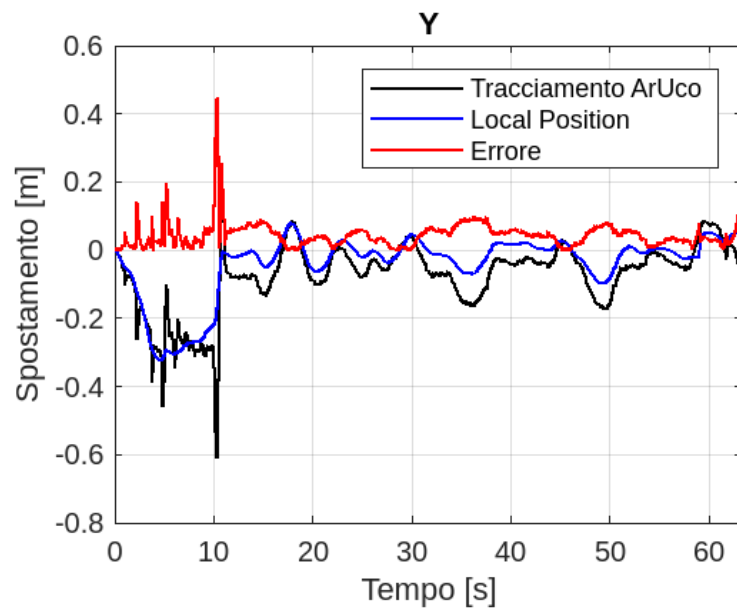


**Figura 60:** Volo con Spin3 – effetto del sensor fusion

Passando quindi al confronto con il tracciamento del marker, vengono forniti nelle figure 61, 62 e 63 i dati relativi agli spostamenti lungo i tre assi.



**Figura 61:** Volo con Spin3 – Spostamento longitudinale

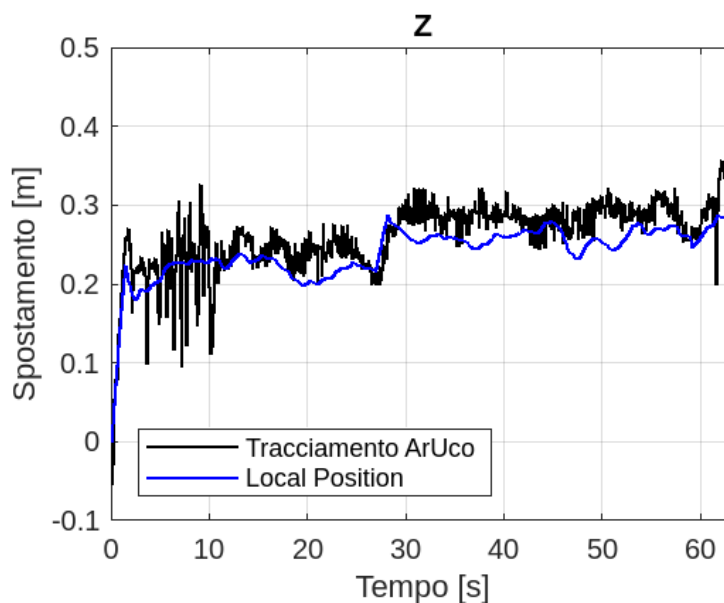


**Figura 62:** Volo con Spin3 – Spostamento laterale

È possibile notare un'ulteriore e notevole diminuzione dell'errore, che risulta rimanere contenuto entro i 10 cm in direzione longitudinale ed addirittura entro i 5 cm in direzione laterale. Anche in questo caso, solo nella fase iniziale del volo i dati di tracciamento manifestano dei picchi dovuti alle variazioni di assetto, ma in seguito l'errore si stabilizza entro gli stretti margini appena indicati. Anche in direzione verticale i dati di tracciamento del marker e di posizione locale coincidono con una soglia di tolleranza centimetrica. Sebbene in questo caso

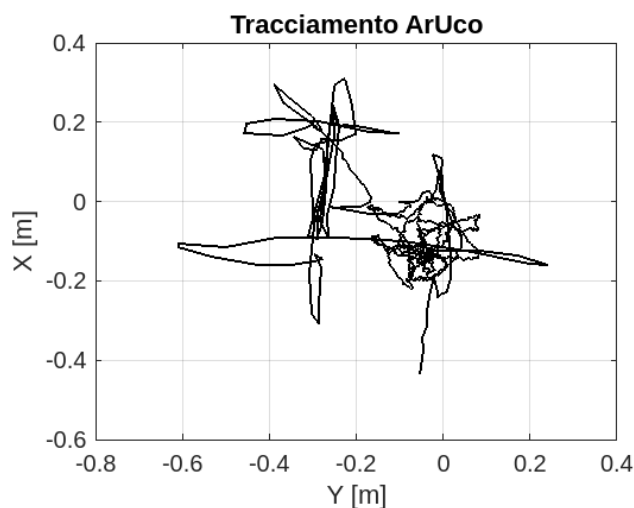


siano omessi i dati del distance sensor, le prove precedenti hanno già dimostrato l'efficacia del marker come riferimento.



**Figura 63:** Volo con Spin3 – Spostamento verticale

Infine, in figura 64 è mostrata la traiettoria nel piano orizzontale ricostruita con i dati di tracciamento del marker. Anche in questo caso si notano gli effetti di alcune manovre iniziali, ma successivamente la traccia ottenuta è naturalmente concorde con quelle basate sui dati GPS e di posizione globale (figura 59).



**Figura 64:** Volo con Spin3 – Spostamenti nel piano (marker)

Riguardo all'utilizzo di correzioni Spin3, è bene aggiungere alcune precisazioni: la funzionalità del servizio si basa sulla connessione ad un server via internet e, in generale come osservabile anche in figura 58, la frequenza di invio dei messaggi di connessione è significativamente inferiore rispetto ad un'antenna

RTK. La combinazione di questi fattori rende il sistema Spin3 meno robusto, tanto che è possibile osservare degli occasionali malfunzionamenti durante il volo. Si veda a tal proposito la figura 65, che fa riferimento ad un ulteriore volo di prova, svolto in preparazione di quello i cui risultati sono appena stati presentati.



**Figura 65:** Instabilità del GPS fix con Spin3

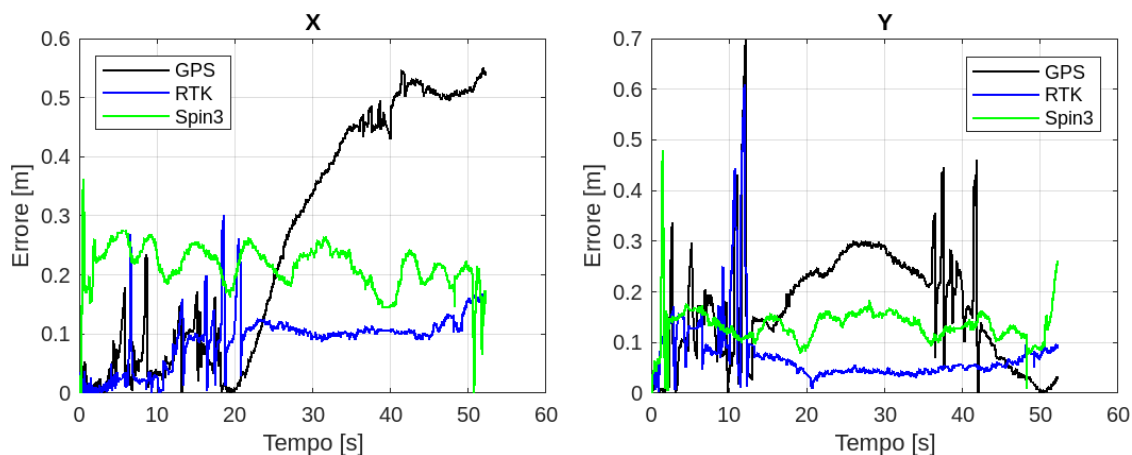
È possibile notare come possano verificarsi dei temporanei decadimenti del livello di GPS fix al livello 5 o, nei casi più gravi, al livello 4, con la perdita totale di correzioni RTK in alcuni istanti. Questo comportamento non è pericoloso, ma causa delle temporanee perdite di precisione nel calcolo della posizione che possono a loro volta causare degli ondeggiamenti del veicolo. Il fenomeno poi è stato riscontrato solo quando si effettuavano manovre più decise e mai in volo stazionario; anche per questa ragione non si ritiene che esso abbia un impatto eccessivamente negativo, ma è certamente importante che gli utenti ne abbiano piena consapevolezza.

#### 4.2.4 Confronti finali

Con il paragrafo conclusivo di questo capitolo si intende completare la presentazione dei risultati con un più diretto confronto delle prestazioni ottenute con i tre voli di prova.

Per prima cosa verranno confrontati gli andamenti degli errori in direzione longitudinale (X) e laterale (Y) per i tre test: la figura 66 mostra la differenza tra i dati GPS e di tracciamento del marker istante per istante per i tre voli sullo stesso grafico. In questo modo è immediato notare come, per altro in linea con le aspettative, l'errore del solo GPS senza correzioni sia il più grande e poi, in particolar modo in direzione X, si può notare come questo subisca degli effetti di drift piuttosto sgradevoli. Con un richiamo alla figura 44 è possibile ricordare

che anche in direzione Y nel primo volo si osservava un drift nella parte finale del volo; in figura 66 ciò è a malapena percettibile a partire da 50 s. Ciò dipende dal fatto che, per sincronizzare i dati dei tre voli in un solo grafico è stato necessario accorciare l'intervallo temporale di osservazione.



**Figura 66:** Errore di posizione lungo le direzioni del piano durante i tre test

Per quanto riguarda le altre due prove, possiamo notare come l'antenna RTK permetta di raggiungere, sebbene in modalità float e non fixed, un errore quantificabile con la metà rispetto all'uso delle correzioni Spin3: infatti, l'errore con Spin3 è mediamente di 20 cm lungo X e 10-15 cm lungo Y e di 10 cm e 5 cm con RTK per le corrispondenti direzioni.

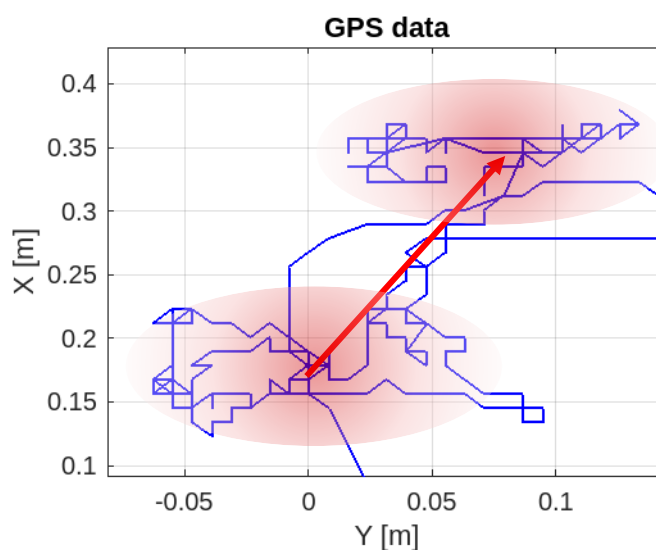
È necessario precisare però che l'errore dell'RTK in direzione X appariva significativamente maggiore in figura 54, con un valore medio di 25-30 cm: a tal proposito si direbbe che le manovre effettuate nella fase iniziale del volo abbiano provocato un errore di offset che si è successivamente mantenuto costante durante il volo. In questo caso si è già detto che l'intervallo di osservazione è stato limitato ed i dati rinormalizzati per permettere un confronto tra le tre prove; presumibilmente con questa procedura di post processing l'errore di offset è stato artificialmente eliminato e si invita quindi a tenere conto anche dei grafici relativi ai singoli voli. La conclusione che è possibile trarre è che l'uso della base RTK permette di ottenere un maggior livello di precisione per applicazioni ad uso locale, ma compiendo manovre più ampie e rapide possono generarsi errori di offset, legati in parte all'accuratezza della posizione della base e forse anche ad un errore di stima nel tracciamento del marker.

L'ultima riflessione si basa sulla capacità del drone di mantenere fissa la propria posizione sulla base delle informazioni di posizione di cui dispone. Si specifica infatti che i tre voli di prova si sono svolti tutti con la medesima modalità: dopo il decollo sono state effettuate delle manovre per posizionare il quadricottero

perfettamente al di sopra del marker per poi lasciare che esso vi si mantenesse autonomamente grazie all'utilizzo della modalità di volo "position". Si richiama ancora il modo in cui questa modalità funziona: quando non vengono impartiti dei comandi, il veicolo mantiene fissa la propria posizione, ma per far ciò si basa sulle informazioni di posizione di cui dispone e dei dati dei sensori. Osservando quindi nel dettaglio la traiettoria del drone durante questa fase di volo a punto fisso è possibile trarre delle conclusioni sull'efficacia delle diverse tecnologie per la determinazione della posizione impiegate nelle varie prove.

Per poter fare ciò, sono stati riportati nelle figure 67, 68 e 69 degli ingrandimenti dei grafici già mostrati nelle figure 48, 52 e 59. Queste nuove figure mostrano quindi ancora la posizione nel piano secondo i dati GPS, ma si concentrano esclusivamente sulla parte di volo a punto fisso al di sopra del marker.

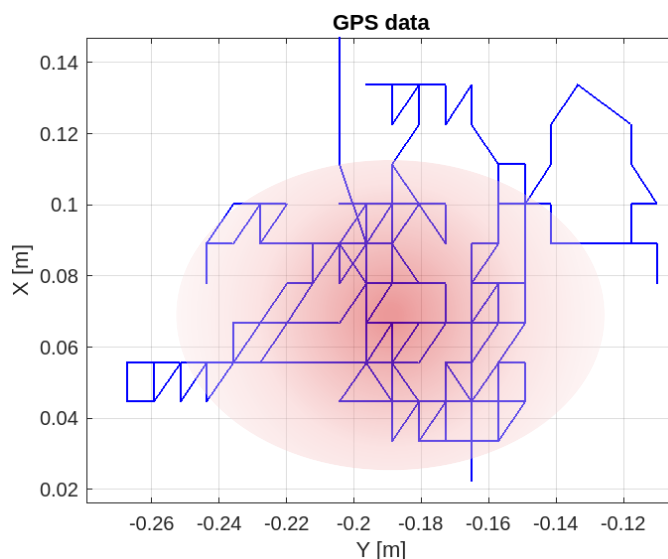
Dalla figura 67 è possibile notare che, utilizzando il solo ricevitore GNSS senza correzioni, la traiettoria è contenuta in un inviluppo di 15 cm x 15 cm circa. Questo risultato è decisamente soddisfacente, ma si deve anche notare che questo sistema è soggetto a drift (indicato con la freccia rossa in figura). La precisione del posizionamento senza correzioni è quindi soddisfacente, ma le prestazioni tendono a decadere rapidamente nel tempo e l'inviluppo della traiettoria, anche se non eccessivamente grande, tende però a spostarsi.



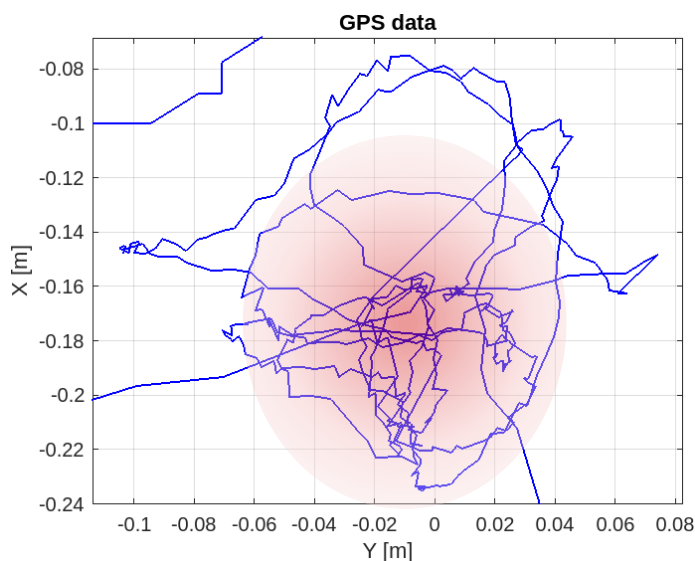
**Figura 67:** Volo con GPS – Traiettoria nel piano durante il volo a punto fisso

Con l'introduzione di sistemi di correzione (RTK in figura 68 e Spin3 in figura 69) viene eliminato il problema del drift e la precisione aumenta: la traiettoria con RTK occupa un'area (YxX) di 10 cm x 7/10 cm, con Spin3 similmente si ottiene una superficie di 10 cm x 6 cm, anche se con alcuni excursus contenibili in una circonferenza di diametro 15 cm. Di fatto sembrerebbe che Spin3 sia un sistema più preciso, ma che possa occasionalmente generare errori più marcati rispetto

alla maggiore costanza del sistema con base RTK, forse proprio a causa della minore frequenza dei messaggi di correzione ed alla loro occasionale mancata accettazione (si veda la figura 58 e relativo commento).



**Figura 68:** Volo con RTK – Traiettoria nel piano durante il volo a punto fisso



**Figura 69:** Volo con Spin3 – Traiettoria nel piano durante il volo a punto fisso

È quindi possibile concludere che i sistemi di correzione permettono di ottenere prestazioni migliori grazie ad un aumento della precisione, ma, almeno per quanto riguarda questi grafici unitamente a quelli presentati nei paragrafi precedenti, non sembra che l'utilizzo di una base RTK o di correzioni Spin3 mostrino notevoli differenze prestazionali; entrambi offrono vantaggi e svantaggi, a proposito dei quali si invita a fare riferimento al capitolo conclusivo.

## 5) Conclusioni

Il primo obiettivo inizialmente posto per questo lavoro di tesi riguardava la configurazione del drone PX4 Vision Autonomy Development Kit: a tale proposito, il veicolo era inizialmente obsoleto e si richiedeva quindi che il suo software venisse aggiornato. Oggi il veicolo è quindi dotato di una versione più recente dell'autopilota, ma soprattutto il suo companion computer, inizialmente privo persino del sistema operativo, è stato configurato in modo da rispondere oggi alle esigenze del gruppo di ricerca nel miglior modo possibile.

A proposito della configurazione del companion computer, la decisione iniziale è stata quella di installare sia ROS che ROS 2 e conseguentemente il sistema operativo Linux Ubuntu 20.04. Di fatto però, durante lo svolgimento dell'attività, si è deciso di abbandonare completamente ROS e completare il setup del framework utilizzando esclusivamente ROS 2. Allora, grazie all'esperienza in tal senso maturata ed alla consapevolezza di poter fare a meno di ROS, la raccomandazione per il futuro è quella di evitare completamente il download di ROS con due vantaggi: il risparmio di tempo ottenuto evitando uno step di configurazione, ma soprattutto la possibilità di installare una versione più recente di Linux e, conseguentemente, anche di ROS 2.

L'obiettivo successivo riguardava il test prestazionale dei sistemi di posizionamento e del marker ArUco come riferimento.

A cominciare dal marker, è possibile concludere che esso rappresenta un riferimento affidabile anche se non estremamente preciso: in primis, si è potuto osservare come i dati di tracciamento del marker per la posizione verticale concordino con le letture del distance sensor a meno di un errore centimetrico (si vedano a tale proposito le figure 45 e 56 del capitolo 4). Anche nelle altre direzioni si è osservata una buona corrispondenza tra i dati di tracciamento ed i dati di posizione provenienti da altre fonti, ma nonostante ciò l'uso del marker ha comunque mostrato alcune criticità: le vibrazioni del drone non filtrate e soprattutto i cambiamenti di assetto rendono i dati di tracciamento notevolmente rumorosi; a tale proposito si potrebbe, per applicazioni future, sperimentare l'implementazione di un algoritmo di filtraggio che possa portare al superamento di queste problematiche o quantomeno che possa ridurre l'impatto.

Per quanto riguarda le prestazioni dei sistemi di determinazione e correzione della posizione del veicolo, è difficile stilare un vero e proprio benchmark in base

alla precisione, soprattutto in mancanza di un ground truth altamente affidabile: l'uso del marker ArUco a tale scopo si è rivelato utile, ma in ogni caso soggetto ad errori. Naturalmente l'uso del GPS senza correzioni, sebbene si sia dimostrato capace di ottime prestazioni, è meno preciso rispetto a quando combinato con sistemi di correzione. Molto più arduo è il confronto diretto tra antenna RTK e correzioni Spin3 dato che hanno generato risultati molto simili, l'unico confronto possibile è quello riguardante pregi e criticità dei due sistemi, in modo che l'utente possa scegliere quello che meglio si adatta alle proprie esigenze.

Per prima cosa, entrambi i sistemi si sono rivelati molto precisi durante la fase di volo a punto fisso, ma producevano un errore più grande quando invece il drone veniva manovrato. In commento ai risultati ottenuti è però necessario rammentare che, con l'uso della base RTK, non è stato possibile raggiungere il livello 6 di GPS fix, probabilmente perché i test sono stati svolti in ambiente urbano e anche perché la base è stata collocata al livello del terreno, non in posizione rialzata. In ogni caso, il sistema RTK si è dimostrato molto più stabile, soprattutto considerando che in alcuni voli è stata osservata la momentanea perdita dei messaggi Spin3: la frequenza delle correzioni RTK è molto più alta e anche per questo la corruzione, perdita o semplicemente l'esclusione di messaggi ricevuti in ritardo ha un impatto significativamente minore rispetto all'uso di Spin3. Per contro, il grosso limite del sistema RTK è quello degli errori di offset ed in particolare la dipendenza dall'accuratezza della posizione della base: queste problematiche giocano un ruolo minore per brevi voli locali, ma possono essere più impattanti se si necessita di conoscere con grande precisione la posizione del veicolo rispetto all'ellissoide di riferimento. Per applicazioni di carattere più "globale" potrebbe essere più adeguato affidarsi al servizio Spin3: la posizione delle sue stazioni è nota con grande precisione e permette quindi un posizionamento accurato non solo a livello locale ma globale, si pensi a tal proposito che questo servizio nasce anche per scopi topografici e cartografici. Si tenga comunque presente che il funzionamento di Spin3 è garantito esclusivamente nelle tre regioni italiane da cui dipende, mentre una base RTK può essere utilizzata ovunque senza dipendere da altri servizi esterni. In conclusione però, per applicazioni che dispongono di una solida connessione alla rete e che si limitano all'area geografica coperta dal servizio Spin3 il suo utilizzo è, con buona probabilità, da preferirsi rispetto ad una base RTK: posto che le due soluzioni sono in buona parte simili, il sistema RTK si basa sull'impiego di attrezzature che necessitano di essere acquistate ad un costo che, specialmente per assicurare elevate prestazioni, può essere significativo, mentre al contrario il servizio Spin3 è completamente gratuito e facilmente accessibile.

## Future work

Le conclusioni cui è stato possibile giungere, unitamente al processo che ad esse ha portato, hanno messo in luce alcune opportunità per futuri sviluppi in questo ambito.

Per quanto riguarda la configurazione del veicolo, è stato messo in evidenza il fatto che l'unica funzione che non è stato possibile implementare, rispetto alla configurazione di fabbrica, è stata quella di obstacle avoidance. Nell'ottica di migliorare e completare il lavoro svolto ed ivi esposto è quindi possibile indagare la possibilità di un'implementazione auto-sviluppata e personalizzata di questa funzione non più ufficialmente supportata dall'autopilota.

Inoltre, per poter più rigorosamente validare l'efficacia del tracciamento del marker ArUco come riferimento, sarà in futuro possibile, in linea con gli obiettivi del gruppo di ricerca, effettuare ulteriori test indoor: l'utilizzo di una gabbia di volo indoor equipaggiata con un sistema Vicon motion capture, offrirà una nuova possibilità di determinazione della posizione del drone con elevatissima precisione ed affidabilità, garantendo così un nuovo termine di confronto. I risultati ottenuti nell'ambito di questa tesi allora, se confermati dalle prove indoor, potranno certificare l'utilizzo del tracciamento di marker come riferimento per le prove outdoor.

Infine, un ultimo sviluppo interessante legato ancora al tracciamento dei marker, potrebbe essere il loro utilizzo non solo per verificare, ma anche per migliorare il calcolo della posizione del veicolo da parte dell'autopilota: posto che, come già anticipato, sarebbe utile provare a sviluppare un algoritmo di filtraggio che elimini il rumore di manovre e vibrazioni dai dati di tracciamento, si potrebbe successivamente inviare tali dati all'autopilota similmente a quanto avviene con i messaggi di correzione RTK e Spin3. In questo modo sarebbe possibile verificare la capacità del drone di mantenere o modificare la propria posizione sfruttando in prima persona i dati di tracciamento dei marker.



## Ringraziamenti

Il più primo e più sentito ringraziamento va in questo contesto al relatore Stefano Primatesta ed al correlatore Riccardo Enrico per la pazienza, i consigli ed il costante supporto; a loro il merito di aver trasformato questo lavoro di tesi in una vera occasione di apprendimento e crescita personale.

Un ringraziamento poi agli insegnanti che, dalle scuole medie all'università, mi hanno condotto lungo il percorso di studi fino a questo momento, che ad esso si pone come finale incoronamento. Ringrazio coloro i quali, con dedizione e consapevolezza, si sono dedicati alla mia formazione con passione, competenza e talvolta con affetto.

Grazie ai miei genitori, per la fiducia e per gli stimoli, ma anche per la libertà ed il supporto, grazie per avermi dato la possibilità e gli strumenti per seguire le mie passioni e costruire la mia vita.

Grazie a mio fratello per non avermi mai disturbato con inutili chiacchiere durante i momenti di studio e di concentrazione, anche se scambiare qualche parola può essere talvolta un piacere.

Grazie anche ai miei amici: quelli nuovi per le occasioni di leggerezza e condivisione durante questo percorso, quelli di lunga data, intimi e sinceri, che in questi anni mi hanno guardato diventare ingegnere, per avermi arricchito ricordandomi di quanto altro potesse esserci.

Infine grazie a Te, che riuscendo ad essere contemporaneamente certezza e scompiglio hai cambiato la mia visione della vita, diventando ulteriore occasione di entusiasmante slancio verso il futuro. Grazie per il supporto e la fiducia, ma soprattutto per ciò che semplicemente sei.

## References

- [1] PX4 Guide - PX4 Vision Autonomy Development Kit  
[https://docs.px4.io/main/en/complete\\_vehicles\\_mc/px4\\_vision\\_kit](https://docs.px4.io/main/en/complete_vehicles_mc/px4_vision_kit)
  
- [2] PX4 Guide - PX4 Vision Autonomy Development Kit/What is Inside  
[https://docs.px4.io/v1.13/en/complete\\_vehicles/px4\\_vision\\_kit.html#what-is-inside](https://docs.px4.io/v1.13/en/complete_vehicles/px4_vision_kit.html#what-is-inside)
  
- [3] PX4 Guide - Path Planning Interface  
[https://docs.px4.io/main/en/computer\\_vision/path\\_planning\\_interface.html](https://docs.px4.io/main/en/computer_vision/path_planning_interface.html)
  
- [4] Holybro Docs - Overview  
<https://docs.holybro.com/drone-development-kit/px4-vision-dev-kit-v1.5/overview>
  
- [5] PX4 Guide - Holybro Pixhawk 6C  
[https://docs.px4.io/main/en/flight\\_controller/pixhawk6c.html#holybro-pixhawk-6c](https://docs.px4.io/main/en/flight_controller/pixhawk6c.html#holybro-pixhawk-6c)
  
- [6] PX4 Guide - Basic Concepts/SD Cards  
[https://docs.px4.io/main/en/getting\\_started/px4\\_basic\\_concepts.html#sd-cards-removable-memory](https://docs.px4.io/main/en/getting_started/px4_basic_concepts.html#sd-cards-removable-memory)
  
- [7] PX4 Guide - Holybro Pixhawk 6C Wiring Quick Start  
[https://docs.px4.io/main/en/assembly/quick\\_start\\_pixhawk6c.html](https://docs.px4.io/main/en/assembly/quick_start_pixhawk6c.html)
  
- [8] PX4 Guide - Assembling a UAS  
<https://docs.px4.io/main/en/assembly/>
  
- [9] PX4 Guide - Power Modules & Power Distribution Boards  
[https://docs.px4.io/main/zh/power\\_module/](https://docs.px4.io/main/zh/power_module/)
  
- [10] PX4 Guide - Companion Computers  
[https://docs.px4.io/main/en/companion\\_computer/](https://docs.px4.io/main/en/companion_computer/)
  
- [11] PX4 Guide - Magnetometer (Compass) Hardware & Setup  
[https://docs.px4.io/main/en/gps\\_compass/magnetometer.html](https://docs.px4.io/main/en/gps_compass/magnetometer.html)
  
- [12] PX4 Guide - Mounting a Compass (or GNSS/Compass)  
[https://docs.px4.io/main/en/assembly/mount\\_gps\\_compass.html](https://docs.px4.io/main/en/assembly/mount_gps_compass.html)
  
- [13] PX4 Guide - Basic Concepts/Buzzer  
[https://docs.px4.io/main/en/getting\\_started/px4\\_basic\\_concepts.html#buzzer](https://docs.px4.io/main/en/getting_started/px4_basic_concepts.html#buzzer)
  
- [14] PX4 Guide - LED Meanings & Tune Meanings  
[https://docs.px4.io/main/en/getting\\_started/led\\_meanings.html#ui-led](https://docs.px4.io/main/en/getting_started/led_meanings.html#ui-led)

[https://docs.px4.io/main/en/getting\\_started/tunes.html](https://docs.px4.io/main/en/getting_started/tunes.html)

[15] Holybro – Structure Core

<https://holybro.com/products/structure-core?srsId=AfmBOoqS2eFhJfJaiTa45aDw6CY9mdig1Yp4RDcPCnpg3f0BMM4pSPp3>

[16] PX4 Guide – Optical Flow

[https://docs.px4.io/main/en/sensor/optical\\_flow.html#optical-flow](https://docs.px4.io/main/en/sensor/optical_flow.html#optical-flow)

[17] PX4 Guide – PMW3901-Based Flow Sensors

<https://docs.px4.io/main/en/sensor/pmw3901.html>

[18] PX4 Guide – Lanbao PSK-CM8JL65-CC5 ToF Infrared Distance Measuring Sensor

[https://docs.px4.io/main/en/sensor/cm8jl65\\_ir\\_distance\\_sensor.html](https://docs.px4.io/main/en/sensor/cm8jl65_ir_distance_sensor.html)

[19] Holybro – Wiring Diagram

<https://docs.holybro.com/drone-development-kit/px4-vision-dev-kit-v1.5/wiring-diagram>

[20] PX4 Guide – PX4 System Architecture

[https://docs.px4.io/main/en/concept/px4\\_systems\\_architecture.html](https://docs.px4.io/main/en/concept/px4_systems_architecture.html)

[21] PX4 vs. ArduPilot: Choosing the Right Open-Source Flight Stack

<https://www.thedroningcompany.com/blog/px4-vs-ardupilot-choosing-the-right-open-source-flight-stack>

[22] PX4 Guide – PX4 Architectural Overview

<https://docs.px4.io/main/en/concept/architecture.html>

[23] QGC Guide – QGroundControl User Guide

<https://docs.qgroundcontrol.com/master/en/qgc-user-guide/index.html>

[24] PX4 Guide – Loading Firmware

<https://docs.px4.io/main/en/config/firmware.html>

[25] ROS – Robot Operating System

<https://www.ros.org/>

[26] ROS 2 Documentation: Foxy – ROS 2 Documentation

<https://docs.ros.org/en/foxy/index.html>

[27] ROS 2 Documentation: Foxy – Recording and playing back data

<https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Recording-And-Playing-Back-Data/Recording-And-Playing-Back-Data.html>

[28] Ubuntu\_20.04 - Install Ubuntu on UP, UP 4000, UP Squared, UP Core, UP Core Plus, UP Xtreme, and UP Squared Pro

[https://github-wiki-see.page/m/up-board/up-community/wiki/Ubuntu\\_20.04](https://github-wiki-see.page/m/up-board/up-community/wiki/Ubuntu_20.04)

[29] ROS 2 Documentation: Foxy – Installation

<https://docs.ros.org/en/foxy/Installation.html>

[30] PX4 Guide – ROS (1) with MAVROS Installation Guide

[https://docs.px4.io/main/en/ros/mavros\\_installation.html](https://docs.px4.io/main/en/ros/mavros_installation.html)

[31] MAVLink Developer Guide

<https://mavlink.io/en/>

[32] PX4 Guide – uXRCE-DDS (PX4-ROS 2/DDS Bridge)

[https://docs.px4.io/main/en/middleware/uxrce\\_dds.html](https://docs.px4.io/main/en/middleware/uxrce_dds.html)

[33] micro-ROS – Micro XRCE-DDS

[https://micro.ros.org/docs/concepts/middleware/Micro\\_XRCE-DDS/](https://micro.ros.org/docs/concepts/middleware/Micro_XRCE-DDS/)

[34] PX4 Guide – Holybro M8N & M9N GPS

[https://docs.px4.io/main/en/gps\\_compass/gps\\_holybro\\_m8n\\_m9n.html#holybro-m8n-m9n-gps](https://docs.px4.io/main/en/gps_compass/gps_holybro_m8n_m9n.html#holybro-m8n-m9n-gps)

[35] PX4 Guide – Holybro H-RTK F9P GNSS

[https://docs.px4.io/main/en/gps\\_compass/rtk\\_gps\\_holybro\\_h-rtk-f9p.html#holybro-h-rtk-f9p-gnss](https://docs.px4.io/main/en/gps_compass/rtk_gps_holybro_h-rtk-f9p.html#holybro-h-rtk-f9p-gnss)

[36] Intel RealSense – Intel RealSense Depth Camera D435i

<https://www.intelrealsense.com/depth-camera-d435i/>

[37] PX4 Guide – Using the ECL EKF

[https://docs.px4.io/main/en/advanced\\_config/tuning\\_the\\_ecl\\_ekf.html](https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html)

[38] PX4 Guide – VehicleGlobalPosition (μORB message)

[https://docs.px4.io/main/en/msg\\_docs/VehicleGlobalPosition.html](https://docs.px4.io/main/en/msg_docs/VehicleGlobalPosition.html)

[39] PX4 Guide – VehicleLocalPosition (μORB message)

[https://docs.px4.io/main/en/msg\\_docs/VehicleLocalPosition.html](https://docs.px4.io/main/en/msg_docs/VehicleLocalPosition.html)

[40] Spin3 GNSS – Il sistema di posizionamento GPS

[https://www.servizi.spingnss.it/spiderweb/ext/guide/Il\\_sistema\\_di\\_posizionamento\\_GPS.pdf](https://www.servizi.spingnss.it/spiderweb/ext/guide/Il_sistema_di_posizionamento_GPS.pdf)

[41] PX4 Guide – Parameter Reference

[https://docs.px4.io/main/en/advanced\\_config/parameter\\_reference.html](https://docs.px4.io/main/en/advanced_config/parameter_reference.html)

[42] PX4 Guide - RTK GNSS (GPS)

[https://docs.px4.io/main/en/gps\\_compass/rtk\\_gps.html](https://docs.px4.io/main/en/gps_compass/rtk_gps.html)

[43] Sciencing - What Is Difference Between RTK Fix & RTK Float?

<https://www.sciencing.com/difference-between-rtk-fix-rtk-float-12245568/>

[44] Spin3 GNSS - Informazioni

<https://www.spingnss.it/informazioni/>

[45] Spin3 GNSS - Stazioni Permanenti

<https://www.spingnss.it/stazioni/>

[46] github - ros\_aruco\_opencv

[https://github.com/fictionlab/ros\\_aruco\\_opencv](https://github.com/fictionlab/ros_aruco_opencv)

[47] Marker creation and detection with ArUco of OpenCV

<https://python-academia.com/en/opencv-aruco/>

[48] ArUco marker detection

[https://clover.coex.tech/en/aruco\\_marker.html](https://clover.coex.tech/en/aruco_marker.html)

[49] PX4 Guide - Flight Modes (Multicopter)

[https://docs.px4.io/main/en/flight\\_modes\\_mc/](https://docs.px4.io/main/en/flight_modes_mc/)