Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Progettazione e Sviluppo di una Piattaforma per la Simulazione e Controllo di un Sistema Multidrone

Relatore:
Chiar.mo Prof.
Angelo Trotta

Presentata da:
Andrea Mancini

Sessione II
Anno Accademico 2020/2021

Abstract

Lo studio e sviluppo di sistemi di controllo per velivoli a controllo remoto (UAV) sono di attualità in ogni ambito, dal sociale al militare e commerciale. Questa tesi si propone di presentare una simulazione il più possibile realistica per governare un swarm virtuale di droni. Nello specifico, ci focalizzeremo nell'implementare ciò utilizzando le librerie messe a disposizione dalla azienda francese Parrot per il controllo dei propri droni commerciali.
Presenterò una architettura software in grado di governare e gestire da uno a più droni utilizzando tre moduli principali:

(i) una applicazione Android, utilizzata come front-end dall'utente, dove sarà possibile pilotare da remoto uno stormo con un ampio ventaglio di features.

(ii) un proxy server in grado di recepire tutte le richieste, elaborarle ed inviare comandi ai droni o messaggi alla applicazione Android.

(iii) un simulatore 3D per creare un mondo virtuale e realistico, dove pilotare i velivoli in sicurezza.

Questo permette varie implementazioni, dal semplice svago, all'utilizzo per scopi militari, ricerca e salvataggio di persone, monitoraggio della flora e della fauna, e più in generale come ausilio nel svolgere quei lavori dove l'intervento umano può essere difficile e rischioso, o addirittura impossibile.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Un aeromobile a pilotaggio remoto, in termini tecnici UAV ("Unmanned Aerial Vehicle"), nasce all'inizio per scopi militari, e tuttora è uno dei campi dove è più usato e dove viene investito più denaro per la sua ricerca e innovazione [1],[2]. Successivamente fu introdotto in ambito civile per le operazioni di ricerca e salvataggio[3], nella sorveglianza aerea delle coltivazioni, nel controllo di linee elettriche e petrolifere, nel monitoraggio della fauna selvatica, in aerofotogrammetria, in riprese cinematografiche, e anche in competizioni di corse [4]. Ultimamente si sta sperimentando già il loro utilizzo per il servizio di delivery, consegna di pacchi, cibo ecc. . . [5]. Noi li conosciamo meglio sotto il nome di droni, nel dettaglio ci riferiamo ai quadricotteri o esacotteri, elicotteri senza pilota a comando remoto con quattro o sei rotori.
Far volare un drone oggi è molto semplice ed alla portata di tutti, ne esistono di tantissimi tipi e marche. Molti di questi sono facilmente controllabili tramite una applicazione scaricabile sul proprio smartphone, da dove è possibile governarli tramite un joystick virtuale disegnato sullo schermo, visualizzare ogni loro parametro, dalla loro altitudine al livello di batteria rimanente, visionare lo streaming della videocamera collegata al velivolo e pianificare dei percorsi. La nuova sfida è controllare più di un drone alla volta, uno stormo intero. Avere a disposizione più droni simultaneamente moltiplica le capacità che si avrebbero con uno solo: si possono svolgere diversi compiti nello

stesso momento, coprire uno spazio aereo più vasto, nell'ambito militare si può parlare anche di un esercito di droni armati, con tutte le controversie riguardo all'etica e alla legalità del caso; in ambito civile permette di cercare e salvare più civili contemporaneamente, e in generale poter svolgere tutti quei compiti elencati precedentemente in scala maggiore per efficacia e potenzialità. Chiaramente la complessità nel gestire e controllare più droni allo stesso momento cresce esponenzialmente. Il problema principale, è evitare la collisione tra i velivoli, specialmente quando i droni si muovono vicini in formazione.

Ad oggi i vari framework per la simulazione messi a disposizione dalle aziende tecnologiche, che costruiscono e commercializzano UAV, non sono sviluppati per la gestione e il controllo di più velivoli simultaneamente. La stessa Parrot non lo supporta ufficialmente, ma permette di eseguire contemporaneamente sull'host più firmware, tanti quanto sono i droni che voglio simulare, mentre Gazebo ne consente la virtualizzazione. Questo ha fatto si che ho potuto sviluppare un framework con una propria architettura per amministrare e controllare uno swarm di droni. E per ottenere ciò mi sono avvalso appunto degli strumenti che Parrot mette a disposizione per la simulazione e il controllo dei propri velivoli. Sphinx e Gazebo per la parte di simulazione e virtualizzazione dei droni, mentre per il controllo di questi Olympe e una applicazione sviluppata per Android. Parlando dell'applicativo Android, questa svolge la parte di front-end di tutto il framework, ovvero lo strumento con cui l'utente si interfaccia direttamente con il sistema. E' un controller per poter svolgere vari casi d'uso, come il movimento in swarm, leader-follow e percorsi pianificati. Esso si interfaccia con Olympe, tramite un proxy server, per inviare i comandi elaborati dall'utente sotto forma di messaggi, sfruttandolo come ponte per governare i droni. Nel capitolo 3 parlerò in dettaglio di questi strumenti e della loro architettura. La potenzialità di questo lavoro cresce grazie alla funzionalità di Olympe di potersi connettere anche a velivoli reali. Questo fa si che dopo una fase di sperimentazione e stabilizzazione in un ambiente simulato, sarà possibile utilizzarlo nella realtà,

aprendo così le porte a tutte quelle applicazioni che ho introdotto a inizio capitolo, dalla sorveglianza di campi agricoli alle operazione di salvataggio. La peculiarità principale è poterlo fare con velivoli commerciali, rendendo ciò più facilmente accessibile a chiunque ne disponesse uno, o più, di questi. Oltre ai droni, gli altri dispositivi che sono indispensabili per questa applicazione sono un computer e uno smartphone Android, che oggigiorno sono comunemente in possesso di ogni individuo, e in particolare di chi possiede un drone, aumentando l'opportunità effettiva di accesso a questa tecnologia e quindi alla possibile platea di consumatori.

L'elaborato sarà strutturato nel modo seguente:

Nel capitolo 2 spiegherò come è strutturato uno swarm, del simulatore che utilizzo e altri presenti in rete, e quali applicazioni smartphone sono disponibili sul mercato.
Nel capitolo 3 presenterò l'architettura del mio framework, i vari moduli di cui è composto e l'implementazione di questi ultimi.
Nel capitolo 4 mostrerò dei casi d'uso con un singolo drone e con molteplici, ed esperimenti su algoritmi di controllo di swarm con i risultati ottenuti.
Nel capitolo 5 concludo il mio elaborato con un accenno a possibili sviluppi futuri.

Capitolo 2

Stato dell'arte

Come descritto nel capitolo precedente, la ragione di questo lavoro è creare un sistema che dia la possibilità all'utente di poter controllare uno swarm di droni simulati in ambienti realistici, utilizzando per lo scopo velivoli commerciali. In chiave futura ci sarà così la possibilità di utilizzare questo framework con droni reali. Per realizzare ciò mi sono documentato sulle varie tipologie di droni disponibili in commercio, cercando di identificarne uno che fosse conciliabile con il mio progetto di simulazione e controllo di uno stormo. In parallelo, trovare un simulatore capace di poter virtualizzare uno swarm di questi droni, e di un SDK compatibile per sviluppare una applicazione Android in grado di governare i nostri velivoli.

In questo capitolo presenterò diverse tipologie di droni e varie strutturazioni e implementazioni di uno swarm , in seguito, due categorie di simulatori e una panoramica sulle applicazioni smartphone presenti sul mercato.

2.1 Droni e swarm

In letteratura vengono studiati e analizzati a fondo gli UAV [4], investigando sulle nuove tecnologie e lo sviluppo in corso di innovativi sistemi, come gli swarm. Di rilievo, le loro applicazioni sul campo civile: sorveglianza dei confini, assistenza nelle emergenze, delivery di beni e costruzioni di

infrastrutture. Di attualità è l'utilizzo per il monitoraggio sull'uso di sistemi di protezione individuali, come le mascherine, a causa della pandemia COVID-19 [6].

2.1.1 Tipologia di droni

I droni possono essere catalogati in varie tipologie, ne esistono di diverse dimensioni e strutture, differiscono per raggio d'azione, per ambito di utilizzo e per gli strumenti con cui sono equipaggiati. In ordine crescente di dimensione ci sono i microdroni, che riproducono per forma e dimensione gli insetti [7]; i minidroni, grandi come il palmo di una mano, seguiti dai droni più convenzionali che pesano dai 500g fino a 30kg, usati professionalmente per riprese cinematografiche, controllo delle colture ecc... [8]; infine abbiamo i droni militari utilizzati dall'esercito[1], di dimensione simili ad aeromobili. Distinguendoli in base alla meccanica abbiamo due macro categorie: gli aeromobili a rotore ed ala fissa. I droni che simulo io rientrano nella categoria a rotore, questi differiscono in base al numero, si parte dal mono rotore fino a otto rotori, octocopter. Dei velivoli venduti al pubblico i più popolari sono i quadricotteri e esacotteri. Le aziende leader del settore sono la DJI [9], la Yuneec [10] e la Parrot [11]. Tutte producono droni muniti di videocamera per lo streaming, di vari sensori tra cui il radar per gli ostacoli; la loro autonomia si aggira tra i 30 minuti a massimo un'ora. Sono anche i più supportati a livello di implementazione software. Ogni drone è progettato per un ambito specifico. Per esempio sulla sorveglianza delle colture, tra le features più all'avanguardia c'è la possibilità di equipaggiare un serbatoio di repellente da spruzzare per la protezione della coltivazione, come sul drone DJI Agras T30 [12]. Il drone che utilizzo nelle mie simulazioni è il modello Anafi prodotto dalla Parrot [13], un quadricottero professionale munito di videocamera stabilizzata, modalità a infrarossi, molti sensori come barometro, magnetometro e ultra-sonar, GPS ecc.... Le differenze sostanziali tra i velivoli con rotori rispetto ad ala fissa sono il decollo e atterraggio verticale e la capacità di volo su un punto fisso (hovering). Con l'avvento di nuove tecno-

logie, come il 5G [14], e i passi in avanti fatti sull'AI, anche i droni si stanno evolvendo beneficiando di queste ultime; infatti si stanno dotando questi velivoli di SIM per la connessione mobile per il controllo a grandi distanze e di riconoscimento degli oggetti tramite l'intelligenza artificiale sia per decidere come comportarsi, sia per creare dei modelli 3D dell'ambiente[13].

2.1.2 Struttura degli swarm

Gli swarm di droni come riporta questo articolo [15] , si possono classificare in diversi modi, uno di questi è suddividerli in base alla loro libertà di muoversi, ovvero stormi totalmente autonomi (fully-autonomous) o parzialmente autonomi (semi-autonomous). Un'altro sistema è immaginare una struttura a strati, layer, dove distinguiamo tra due possibili modalità:

1. single-layered dove ogni drone è il leader di se stesso

2. multi-layered dove ogni drone dipende dai leader appartenenti a quel livello, i quali a loro volta fanno capo ai leader del layer gerarchicamente più alto. In questo sistema il computer che funge da server-station si trova al livello più importante della gerarchia. I velivoli all'interno dello stormo possono avere una collezione di dati dedicata ed eseguire task in real-time; mentre i processi più pesanti e dispendiosi vengono eseguiti dalla server-station.

Il mio swarm si può adattare a due situazioni. La prima è a single-layered, ciò significa che tutti droni ricevono lo stesso comando dalla server-station centrale. La seconda condivide sia le caratteristiche di semi-autonomia sia quelle multi-layered. In dettaglio esisitono due strati, in uno è presente un leader che coordina un cluster di droni subordinati; il leader a sua volta dipende dalla server-station allo strato sovrastante. La semi-autonomia è data dall'unica parte automatica che è l'algoritmo per evitare la collisione e il mantenimento della formazione a stormo.

2.2 Simulatori grafici droni

Riguardo ai simulatori possiamo distinguerli facilmente dal proposito per cui sono stati creati.

2.2.1 Simulatori per skill-training

Da una parte abbiamo gli emulatori costruiti per ricreare fedelmente l'esperienza di volo, con l'obbiettivo principale di permettere all'utente di allenarsi e prendere confidenza con il velivo in un ambiente sicuro. In questo modo il pilota può simulare vari casi d'uso, tra cui aggirare ostacoli, seguire un percorso, affrontare differenti tipologie di ambiente con condizioni climatiche diverse. Un esempio è il simulatore della DJI [16]. Al suo interno troviamo chiaramente solo droni della azienda; inoltre propone varie situazioni di volo per fare pratica e raffinare le proprie skill, collegando eventualmente il proprio controller remoto DJI. Più aperto come simulatore troviamo Zephyr Drone Flight Simulator [17], costruito appositamento per accademie di addestramento e scuole; virtualizza differenti tipologie di droni e di diverse marche. Un ultimo emulatore è RealFlight Simulator [18], è un simulatore a controllo radio. Il software permette di pilotare diversi aeromobili, elicotteri e droni. L'utente può imparare a governarli, fare pratica e anche volare in modalità multigiocatore. La particolarità di questo programma è l'integrazione di un editor per creare nuovi aeroporti oppure per cambiare la fisica e la aerodinamica di velivoli già esistenti; in aggiunta con Autodesk 3Ds[19] si possono creare anche nuovi modelli; supporta anche l'integrazione con SITL.

Tutti questi simulatori differentemente da quelli che introdurrò nel prossimo paragrafo, sono progettati per il commercio e la vendita, e di tipo closed-source. Non permettono tutte quelle operazioni di sperimentazione di sistemi, integrazione e implementazione necessarie per il mio progetto. Principalmente perché non sono stati pensati per lo scopo, e per questo non si ha la possibilità di virtualizzare e controllare più di un drone nello stesso momento.

2.2.2 Simulatori robots

In letteratura [20] vengono presentati gli emulatori più importanti sviluppati per la ricerca e implementazione di sistemi robotici, tra cui i droni. Tra i simulatori più famosi ci sono Gazebo, WeBots, V-REP, ARGoS e AirSim. SwarmLab viene presentato nell'articolo citato sopra.

Gazebo [21], è il simulatore che utilizzo in questo lavoro. Completamente open-source, integra al suo interno il motore fisico ODE, il rendering tramite OpenGL, e supporta la simulazione di sensori e actuator control, può riprodurre una grande quantità di tipologie di robot, elementi fisici ambientali come edifici, alberi, strade e acqua. E' possibile tramite dei plugin di integrarlo con ROS e SITL[22],[23]. Viene utilizzato come simulatore in varie competizioni tecnologiche organizzate da aziende ed enti come l'americana NASA e la giapponese Toyota.

WeBots [24], open-source, utilizza lo stesso motore fisico di Gazebo, ma fornisce le proprie API per un maggior numero di linguaggi di programmazione e nativamente include modelli di droni.

V-REP [25] simulator invece, ora sotto il nome di CoppeliaSim, offre funzionalità per facilitare l'editing di robot e altri modelli. Lo sviluppo di queste può essere fatto tramite Lua, un interprete built-in o usando le sue API in C o Python.

Spostandoci maggiormente verso lo swarm robotics, abbiamo ARGoS e CUSCUS. ARGoS [26] rappresenta un alternativa semplice che offre un buon compromesso tra scalabilità e estensibilità. Infatti permette di simulare un grande numero di robots e la possibilità di usare differenti motori fisici, ma non include modelli di droni nativamente.

CUSCUS [27] è un simulatore che nasce per la gestione di protocolli di rete, per la comunicazione e controllo di velivoli, con l'obbiettivo di governare una flotta di UAV. In particolare integra al suo interno due strumenti: FL-AIR un framework che si occupa del controllo degli UAV e NS-3 che simula la gestione dei protocolli di rete. Scenario Module si concentra sulla modellazione 3D importando scenari creati direttamente da OpenStreetMaps,

realizzando così un ambiente tridimensionale e realistico dove i droni possono muoversi e sperimentare vari casi d'uso. Disegnato specificatamente per droni e automobili, AirSim [28] è un simulatore più recente costruito su Unreal Engine, e come Gazebo, permette l'integrazione di SITL e di un controllore di volo come PX4. In AirSim, le simulazione multi-agents sono facile da mettere a punto, e funzionalità custom possono essere scritte in C++ e Python. SwarmLab è un simulatore costruito appositamente per studiare il comportamento di singoli droni e swarm. Al suo interno Matlab viene utilizzato per implementare e fare debug di algoritmi rapidamente, con l'ausilio di un grande database di funzionalità built-in, realizzando grafici e video con il minimo sforzo. E' capace di virtualizzare uno stormo composto da centinaia di velivoli. Manca di modellazione 3D e di ambienti realistici, principalmente perché il sistema è stato pensato per lo scopo puramente accademico.

La quasi totalità di questi simulatori sono basati su potenti motori di rendering 3D, e sono per lo più scritti in C++, di conseguenza la grafica è molto realistica. Il punto debole è l'obbligo di conoscenza di più di un linguaggio di programmazione.

2.3 Applicazioni smartphone controllo droni

L'innovazione nel campo degli smartphone e degli UAV cresce di pari passo costantemente ed a grande velocità. Inevitabilmente questi due mondi si sono incrociati, per coniugare la portabilità e facilità d'uso dei dispositivi mobili con l'esperienza di volo di un drone [29]. Sui due sistemi operativi principali sono presenti le maggiori applicazioni pubblicate dai brand più importanti [30], [31]. Sia DJI che Parrot hanno sviluppato numerose app per controllare e gestire i propri droni con importanti e avanzate funzionalità: Definire modalità e piani di volo, punti di interesse, chiamati POI, da osservare durante il percorso, accesso ai dati del drone come il livello di batteria e la sua posizione geografica, foto e video catturati durante le missioni, oltre al videostreaming in real-time. Ma esiste una differenza sostanziale tra

le loro app: DJI non supporta il controllo manuale tramite un joystick virtuale all'interno della propria applicazione. Per comandare i loro droni è necessario collegarsi dal proprio dispositivo mobile all'access point creato dal loro controller, perché è tramite esso che l'app riceve le informazioni come le statistiche del drone e il videostreaming. Il collegamento tra il velivolo e il controller è via radio. Diversamente, l'azienda francese ha sviluppato varie applicazioni con controllo manuale, come FreeFlight. I droni Parrot come Anafi possiedono il proprio access point a cui l'applicazione si connette direttamente. Parrot mette a disposizione un SDK per entrambe le piattaforme Android e iOS, per permettere di sviluppare a tutti applicazioni utilizzando i loro strumenti e le loro API.

Capitolo 3

Modellazione ed Implementazione

3.1 Struttura della architettura

In questo capitolo presenterò l'architettura del nostro framework. In che modo i suoi componenti si interfacciano tra di loro, quali tecnologie utilizzano, e il ruolo che svolgono all'interno del sistema.

Nella sezione 3.1 introdurrò i vari componenti e un grafico per visualizzare meglio come è composto l'insieme.

Nella sezione 3.2 parlerò più in profondità dei meccanismi che governano il framework, dell'implementazione di ogni modulo e di come avviene la comunicazione tra di loro.

3.1.1 Architettura

L'architettura è composta da:

1. un dispositivo mobile con installata l'app Android
2. una macchina Linux con a bordo il software Python che integra Olympe
3. il simulatore Gazebo sulla macchina Linux

4. interfacce VLANs create da Sphinx per ogni drone simulato

5. due server per la comunicazione tra i dispositivi

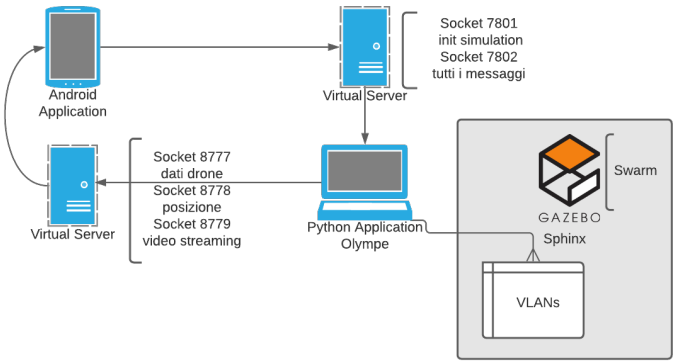

Figura 3.1: Architettura Framework

La scelta nell'usare come protocollo di rete Transfer Control Protocol, per comunicare tra i due dispostivi, è stata dettata dalla necessità di avere maggiore garanzia nella consegna dei pacchetti insieme alla velocità elevata, in particolare modo nell'invio dei comandi per il controllo manuale.

Per utilizzare una comunicazione efficiente e comune ai due sistemi ho scelto di usare il formato javaScript Object Notation, utilizzato spesso tra applicazioni client e server. Un esempio di un messaggio inviato da Android al ProxyServer per il comando manuale è mostrato in figura 3.2.

3.1.2 Componenti

Sphinx

Sphinx è un tool distribuito da Parrot per la simulazione dei droni dell'azienda francese, pensato per aiutare lo sviluppo del software. Esegue il firmware del drone scelto per la simulazione in un ambiente separato rispetto all'host. Utilizza come simulatore Gazebo, dove viene caricata l'anatomia del velivolo, 'my.drone' un file XML, con all'interno tutte le sue impostazioni.

```
JSONObject message = new JSONObject();
try {
     message.put( name: "type",  value: "PCMD")
            .put( name: "x", x)
            .put( name: "y", y)
            .put( name: "height", height_drone)
            .put( name: "rotation", rotation);


} catch (JSONException e) {
    e.printStackTrace();
}
```

Figura 3.2: Esempio json per comando manuale

Tra queste la "⟨stolen-interface⟩" che, se attivata, permette al firmware di utilizzare l'interfaccia di rete WiFi come Access Point (AP). In questo modo il drone simulato è controllabile dai controller esterni all'host, come le applicazioni mobili che utilizzano GroundSDK e anche tramite lo Skycontroller, sviluppati da Parrot. Questo perché è possibile connettersi al drone tramite tre possibile link fisici, il cui supporto cambia in base al modello. Un esempio in Tabella 3.1.

Oltre a questa impostazione, abbiamo i settaggi per attivare o disattivare la videocamera, lo stabilizzatore gimbal, simulare l'utilizzo di una SD card, ecc... Elencando altre funzioni importanti che svolge Sphinx abbiamo la possibilità di modificare il comportamento del drone tramite una interfaccia Web o linea di comando. Con comportamento del drone si intende andare a manipolare i sensori come il GPS, i gruppi di propulsione del velivolo e la sua aerodinamica. In seguito è disponibile lo streaming della videocamera del velivolo e la possibilità di essere eseguito in remoto, creando un servizio di client e server su due macchine. Può essere installato solo su sistemi Linux a 64 bits. Il drone che abbiamo utilizzato noi è il modello Anafi4k in Tabella

drone	virtual ethernet	wifi	bluetooth
Airborne	no	no	yes
Anafi4k	yes	yes	yes
Bepop	yes	yes	no
Bepop2	yes	yes	no
Disco	yes	yes	no
Mambo	no	no	yes
Swing	no	no	yes
Bluegrass	yes	yes	no

Tabella 3.1: Modelli

3.1. La sua scelta è stata dettata dalla capacità di potersi interfacciare con esso attraverso tutti e tre i link fisici introdotti prima. Oltre a essere l'ultimo modello prodotto dalla casa francese, è il più supportato a livello di aggiornamenti software ed è equipaggiato con le ultime tecnologie, la videocamera ad infrarossi e la funzionalità cameraman, che permette al drone di seguire un punto di interesse, come un oggetto in movimento.
Per eseguire Sphinx, l'utente eseguirà l'apposito comando dal terminale della macchina:

```
sphinx PATHTO/sphinx/drones/anafi4k.drone
```

All'avvio dell'esecuzione verranno create diverse sottoreti. Per ogni drone simulato verrano inizializzate due Virtual Ethernet: una sull'host con indirizzo IP 10.202.x.254 chiamata "fdvethx", e una sul drone simulato identificandola "ethx" all'indirizzo IP 10.202.x.1 con $0 \leq x < n$, n numero dei droni da simulare. I droni saranno così tutti accessibili e distinti attraverso la propria sottorete, associata a quella presente sull'host. Nel progetto ho scelto di utilizzare proprio questo tipo di link per connetterci al drone, limitandomi però a dovere eseguire il controller in locale sulla macchina. Questo passaggio lo svilupperò nel paragrafo 3.2.

Olympe

Il modulo che utilizza Olympe svolge tutta la parte di back-end del framework, lo strumento sui cui si appoggia l'applicativo Android per pilotare i droni. E' stato costruito come gestore di tutte le richieste in ricezione dall'app. Al suo interno è composto di un server per ricevere i messaggi, di un gestore associato al controller per l'esecuzione delle richieste, come il pilotaggio manuale e l'avvio di percorsi pianificati, e di un client TCP per inviare dati utili all'utente sul proprio dispositivo mobile.
Olympe è un framework Parrot che fornisce una interfaccia programmabile in Python per il controllo dei droni della casa francese. Viene utilizzato principalmente insieme a Sphinx come controller per i droni simulati. Tramite uno script eseguito sulla propria macchina, ci si può connettere al drone sia simulato che fisico e controllarlo tramite le API fornite. É costruito sulla architettura "arsdk-ng/arsdk-xml", comune agli altri framework dell'azienda per Android e iOS. Le funzioni principali sui cui si basa Olympe sono l'invio di messaggi di comando al drone come pilotaggio, orientamento della camera, FlightPlan, ricevere lo stato corrente del drone come le sue impostazioni; controllo e registrazione del video streaming con metadata associato. Anch'esso è disponibile solo su sistemi operativi Linux. In figura 3.3 e 3.4 un esempio sull'uso e funzionamento delle API di Olympe per muovere un drone in avanti di dieci metri.

```
1  # -*- coding: UTF-8 -*-
2
3  import olympe
4  from olympe.messages.ardrone3.Piloting import TakeOff, moveBy, Landing
5
6  DRONE_IP = "10.202.0.1"
7
8  if __name__ == "__main__":
9      drone = olympe.Drone(DRONE_IP)
10     drone.connect()
11     assert drone(TakeOff()).wait().success()
12     drone(moveBy(10, 0, 0, 0)).wait()
13     assert drone(Landing()).wait().success()
14     drone.disconnect()
```

Figura 3.3: Esempio del comando moveBy di Olympe

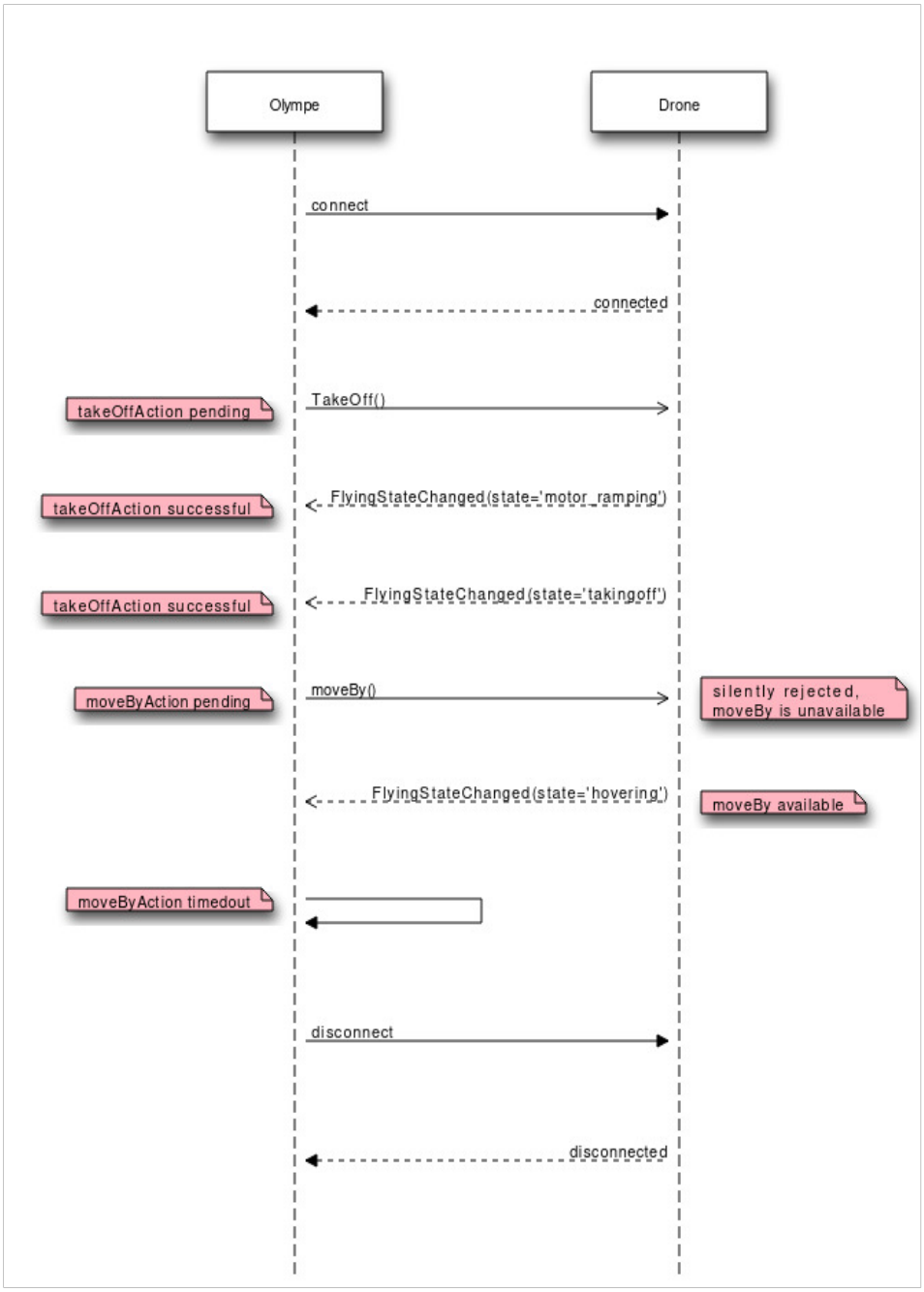

Figura 3.4: Diagramma comunicazione tra Olympe e il drone

Android

L'applicazione è stata concepita per aumentare la portabilità del framework a una situazione realistica di controllo dello swarm utilizzando uno

smartphone o tablet, avvicinandoci alle soluzioni già presenti oggi di pilotaggio tramite dispositivi mobili. E' stato scelto di utilizzare Android per la sua grande diffusione nei dispositivi mobili. Inoltre permette una maggiore flessibilità nello sviluppo, per le sue basse restrizioni a livello di environment.

L'applicazione è scritta in Java, compilata alla versione SDK 30 mentre 24 è la versione minima. Utilizza i servizi di GoogleMaps, per visualizzare la posizione dei droni in real-time e la creazione di percorsi personalizzati, mentre Firebase per lo storage dei video e la visione in streaming di questi ultimi. L'orientazione è orizzontale, obbligata dal corretto utilizzo dei joystick e per la visione dello streaming video.
Nel paragrafo 3.2.3 illustrerò la struttura dell'applicazione e il funzionamento di ogni suo modulo.

3.2 Integrazione ed Implementazione

3.2.1 Streaming Video

Nella realizzazione del video streaming sono state percorse numerose strade. Due modalità esistono per accedere allo streaming della videocamera frontale:

(i) Utilizzando Parrot PDraw, un framework creato apposta per visualizzare i media dei droni come Anafi. Supporta sia lo streaming RTP/RTSP sia il formato MP4. Olympe integra al suo interno PDraw, mediante il quale apre e crea uno streaming video all'indirizzo "rtsp://ipdrone/live". A causa del fatto che i nostri droni sono accessibili solo tramite la vlan associata al drone, l'ip è locale alla macchina. Ho provato così ad utilizzare VLC per creare un server rtsp accessibile all'interno della rete locale, utilizzando l'ip della macchina, con sorgente lo streaming creato da Pdraw. Purtroppo questa soluzione non era efficiente in termini di prestazioni. Il video era in ritardo e aveva problemi di buffering.

(ii) Olympe fornisce API per gestire lo streaming. Come "setstreamingcallbacks" che permette di associare al drone delle funzioni che vengono richiamate ogni volta che un nuovo frame è disponibile, codificato in H.264 MainProfile o in YUV420. Purtroppo in questo passaggio non sono riuscito a decodificare i frame H.264 una volta ottenuti sull'applicazione Android. Ho provato così con il formato YUV, decodificandolo in RGB utilizzando lo schema in figura 3.5.

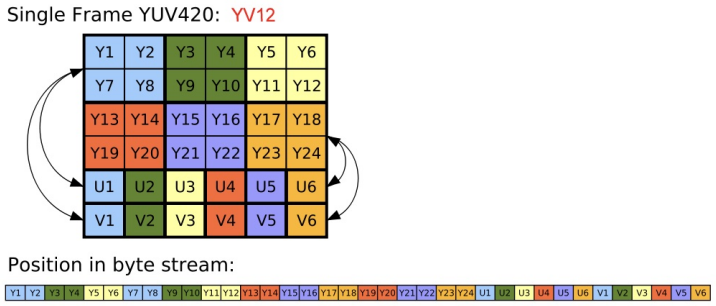

Figura 3.5: YUV schema

L'operazione ha avuto successo ma le prestazioni sono inefficienti, in termini di dimensioni la differenza per frame tra YUV e H.264 è superiore a un milione di bit in favore di quest'ultimo. Questo causa un ovvio ritardo tra il video visibile sullo schermo e l'effettiva situazione del drone, oltre al carico computazionale nella decodifica.

3.2.2 Modulo Olympe

Per ovviare al problema degli Access Point, è stato scelto di utilizzare Olympe come controller principale del sistema. Permette di connettersi ai droni soltanto specificando l'IP di questi, dal momento che possiedono ognuno la propria VLAN, interfaccia creata al lancio di Sphinx. Il software che utilizza Olympe, scritto in Python, è il punto cardine di tutta l'architettura. Al suo interno è composto dai seguenti moduli:

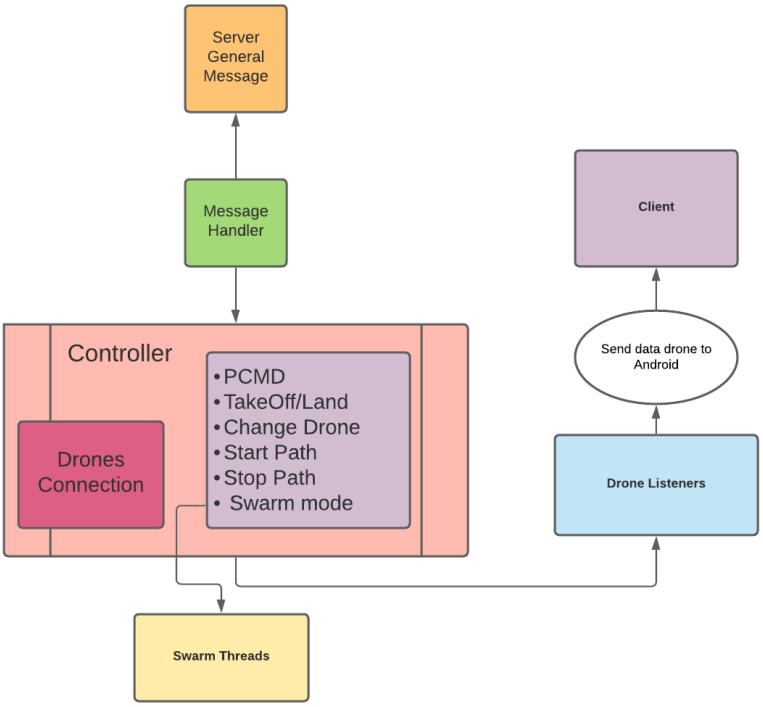

Figura 3.6: Olympe Architecture

1. il Server riceve tramite la rete pacchetti inviati con protocollo TCP, dal dispositivo mobile dove viene eseguita la applicazione Android. I pacchetti che si scambiano i due dispositivi sono messaggi composti da oggetti JSON. Le porte utilizzate dai socket del server sono la 7801 e la 7802. La prima viene utilizzata solo una volta per ricevere il pacchetto contenente le informazioni per avviare la simulazione (numero dei droni e formazione di spawn) utilizzando Sphinx. Poi il socket viene chiuso. La seconda per tutti i messaggi seguenti: comando manuale, TakeOff e Land, percorso pianificato, cambio del leader, interruzione del percor-

so. Tutte queste features le discuterò nel prossimo sottoparagrafo che riguarda il controller.

2. il Client invia, utilizzando lo stesso metodo, i dati ricevuti dai listeners, creati per ogni drone simulato, al server sullo smartphone. In base al tipo di dato ci sono tre socket dedicati:

   (i) Socket 8777, vengono condivisi i dati su altimetria, attitude e flying-state per monitorare il suo stato corrente (hovering, landed, flying ecc. . .) .

   (ii) Socket 8778, per la posizione geografica compresa l'altitudine.

   (iii) Socket 8779, per la trasmissione dello streaming video. Ogni frame ha dimensione 1382400 bytes, per questo viene spezzato in 22 pacchetti da 65535 bytes, dimensione massima supportata dal protocollo TCP.

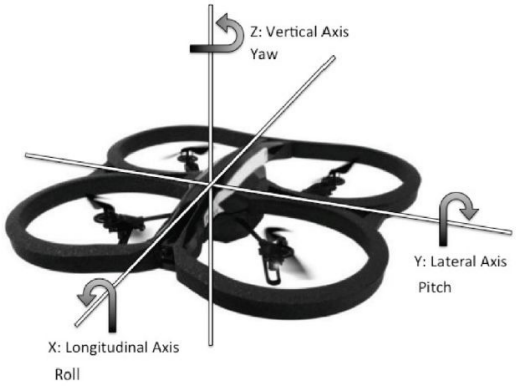

Figura 3.7: Assi di movimento del drone

3. Controller, in base al tipo di messaggio ricevuto vengono eseguite le relative funzioni:

   (i) *pcmd*, è il comando per controllare manualmente i droni tramite i valori generati dai joystick di pitch, roll, yaw e gaz come in figura3.7

(ii) *changedrone*, ricevuto questo messaggio viene cambiato il drone che si sta seguendo con quello indicato nell'oggetto Json. Il video streaming verrà reinizializzato sul drone, come i parametri di altimetria e attitude visualizzati sullo smartphone che corrisponderanno alla nuova scelta.

(iii) *take off*, è il comando per il decollo dei droni. All'inizio della simulazione verrà fatto decollare tutto lo swarm, un componente alla volta per controllare che sarà eseguito correttamente. In seguito solo il drone che si sta seguendo.

(iv) *land*, comando per l'atterraggio dei droni. Controllato che il drone seguito è in stato di hovering, condizione necessaria, verrà fatto atterrare.

(v) *startpath*, ricevuto questo messaggio una percorso pianificato partirà utilizzando i droni e i punti geografici indicati nel json .

(vi) *stoppath*, interrompe il percorso che stanno eseguendo i droni indicati, se ne esiste uno.

(vii) *swarm mode*, con questo messaggio viene attivata la modalità a stormo. il drone che si sta seguendo diventerà il leader dello swarm, gli altri velivoli saranno subordinati a lui attraverso l'uso di un algoritmo. Così facendo manterrano la formazione e seguiranno il leader.

3.2.3 Applicazione Android

Per la realizzazione ho pensato di usare GroundSDK, framework messo a disposizione da Parrot per sviluppare app sia iOS che Android, utilizzando le API costruite sull'architettura "arsdk-ng/arsdk-xml" comune a tutti i loro sistemi. FreeFlight, sullo stesso framework, si collega all'AP corrispondente al drone simulato con Sphinx. Dopo varie ricerche e tentativi, a causa proprio del meccanismo di creazione dell'access point per potersi connettere al drone, l'utilizzo dell'SDK per la nostra app non era compatibile con il nostro

lavoro. Infatti avremmo dovuto creare molteplici AP, uno per ogni drone, e questo non è possibile perché la scheda di rete montata sul maggior numero di macchine è soltanto una. Una soluzione poco pratica e difficilmente realizzabile è collegare tramite usb le schede di rete necessarie, ma questo è limitato dal numero di porte disponibili sulla macchina. Dall'altro lato il dispositivo mobile avrebbe poi dovuto gestire la connessione multipla a più access point. Del kit di sviluppo ho utilizzato soltanto tre sue componenti, JoystickView, AttitudeView e AltimeterView; utili per pilotare il drone, e visualizzare la sua altitudine e il suo assetto.

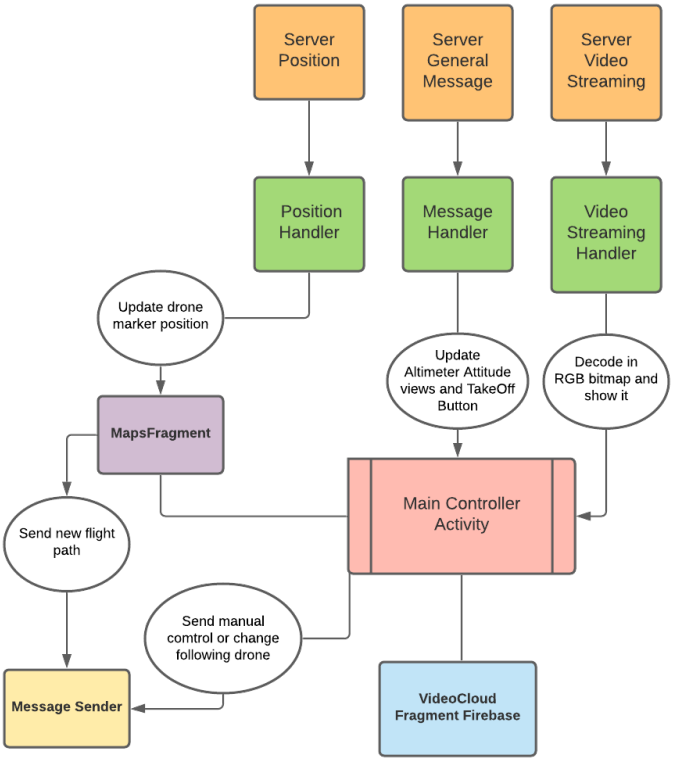

Figura 3.8: Architettura applicazione Android

1. VideoCloud Fragment. Dalla activity principale è possibile accedere a questo fragment per la visione dei contenuti video caricati su Firebase. Firebase è una piattaforma di Google per la creazione di applicazioni

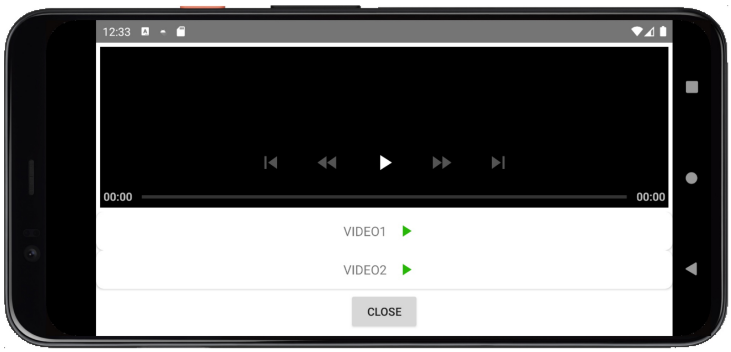

Figura 3.9: VideoCloud Fragment Layout

mobili. Degli strumenti di cui dispone, noi utilizziamo lo storage di file e il database. La visualizzazione in streaming dei contenuti avviene ottenendo dal database l'url corrispondente al video caricato sulla piattaforma, tramite le relative API.

2. GoogleMaps Fragment:

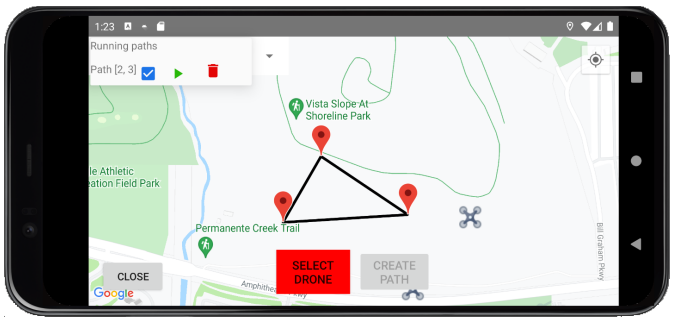

Figura 3.10: MapsFragment Layout

Per la creazione di percorsi pianificati, ho pensato che il metodo migliore per l'utente di scegliere i vari punti del tracciato fosse interagire con una mappa. Per questo scopo ho sfruttato i servizi di GoogleMaps. Oltre alla creazione di un path, la mappa permette di visionare

la posizione dei droni indicati da un marker apposito. L'utente, per pianificare il percorso, deve svolgere le seguenti operazioni :

(i) Selezione dei droni interagendo con il loro marker sulla mappa.

(ii) Premuto il bottone "create path", creare i punti del percorso cliccando liberamente sulla mappa.

(iii) Con gli appositi bottoni, l'utente potrà completare l'operazione o annullarla.

I percorsi creati saranno visibili su una dropdown list. Interagendo con essa, si potrà :

(i) Inviare il path al server sulla macchina Linux per comunicarne l'avvio.

(ii) Interrompere il path.

(iii) Mostrare o nascondere il percorso sulla mappa.

(iv) Eliminare il percorso.

3. MainController Activity:

Figura 3.11: MainController Activity Layout

Il controller principale si trova in questo modulo. Da qui, si può accedere a tutte le features disponibili, interagendo con gli oggetti che lo popolano :

(i) I due joystick, uno a sinistra e l'altro a destra dello schermo, servono per il controllo manuale dei droni. Il primo genera i valori di pitch e roll, il secondo di yaw e gaz.

(ii) Altimeter e Attitude misurano, tramite i dati ricevuti dai droni, rispettivamente l'altitudine e l'assetto di questi.

(iii) ChangeDrone è la lista dei droni simulati. Tutti i dati ricevuti da Olympe corrisponderanno al velivolo selezionato in questa lista.

(iv) I bottoni "maps" e "video on cloud" permettono l'accesso ai relativi fragment.

(v) Il toggle per attivare la modalità swarm, Olympe creerà per ogni drone un thread dedicato dove eseguire l'algoritmo di gestione.

4. MessageSender è il modulo per comunicare tramite protoccolo TCP con Olympe. Viene utilizzato da tutti i moduli che necessitano di inviare gli oggetti JSON generati.

5. Server e Handler:

   (i) Server Position riceve da Olympe la posizione aggiornata di tutti i droni in tempo reale, Position Handler si occupa di aggiornare la posizione dei marker dei droni sulla mappa.

   (ii) Server General Message riceve tutte le altre comunicazioni riguardo lo stato dei droni, Message Handler controllando il tipo del messaggio decide l'operazione da svolgere. Es.: Aggiornare l'altitudine

   (iii) Server VideoStreaming, ricevuti tutti i 22 pacchetti, l'Handler specifico si occuperà di convertirli in RGB, generando una Bitmap per la creazione di un frame da mostrare sul MainController Activity.

Capitolo 4

Valutazione

In questo capitolo presenterò due modalità di utilizzo del framework, simulando un singolo drone o molteplici. In seguito verranno proposti tre esperimenti svolti per l'analisi del comportamento dei velivoli in uno swarm, sotto la gestione di un algoritmo.

4.1 Singolo drone

In figura 4.1 lo schema del funzionamento del framework controllando un solo drone. L'applicativo Android comunica con il server sulla macchina. Invia principalmente i comandi manuali generati dai joystick presenti in figura 4.1 . Come spiegato nel capitolo 3, Olympe si connette al drone attraverso la VLAN creata da sphinx al lancio della simulazione. Il controller all'interno della applicazione Python elabora i messaggi ricevuti utilizzando la funzione apposita di Olympe per il pilotaggio manuale: PCMD. I dati che vengono utilizzati sono il pitch, roll, yaw e gaz; muovono il drone su tre assi perpendicolari come spiegato in figura 3.7 . La stessa procedura avviene per gli altri messaggi:

1. Premendo il bottone di TakeOff/Landed, il controller utilizza le funzioni apposite di TakeOff e Landed

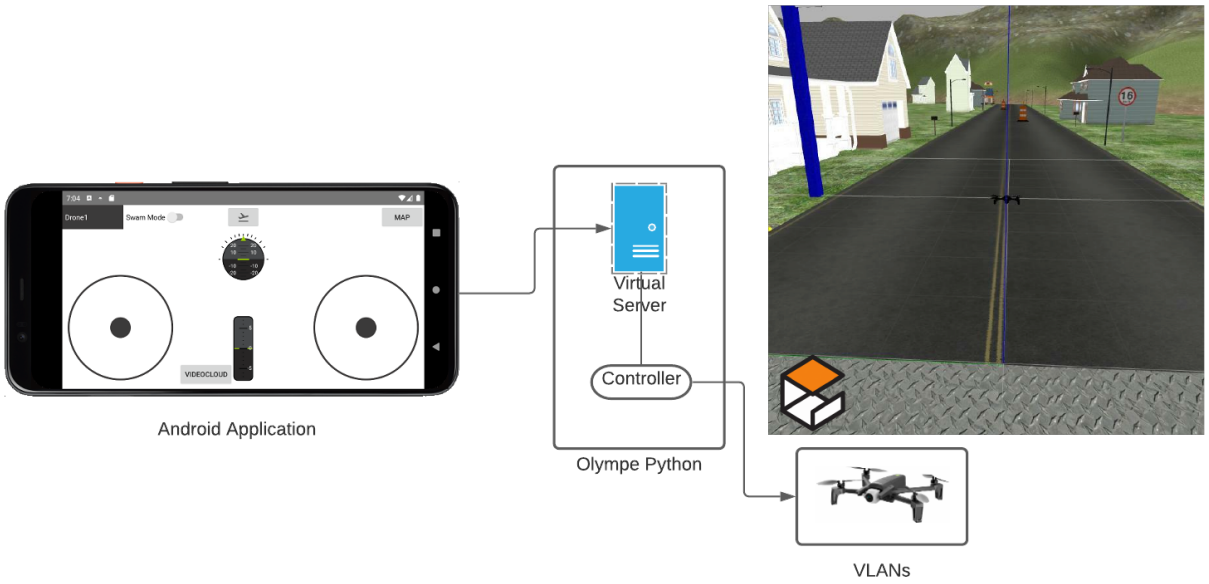

Figura 4.1: Schema Singolo drone

2. Creato e inviato il percorso sfruttando GoogleMaps sull'app Android, viene creato un thread dove per ogni punto geografico del percorso viene usata la funzione moveTo fornita da Olympe, controllando il passaggio del drone per ognuno di questi.

L'utente sarà così in grado di svolgere tutte le operazioni elementari per pilotare il proprio drone, monitorando il tutto sul simulatore Gazebo per un'esperienza realistica di volo.

4.2 Multi drone

A differenza del singolo drone, il procedimento di settaggio della simulazione sull'applicazione Android è fondamentale. Come in figura 4.2 vengono chieste all'utente di indicare il numero dei droni da simulare e la loro formazione di spawn scegliendo tra: inline, single-line, triangle, square. Il controller elabora i messaggi ricevuti considerando i droni indicati per l'esecuzione del-

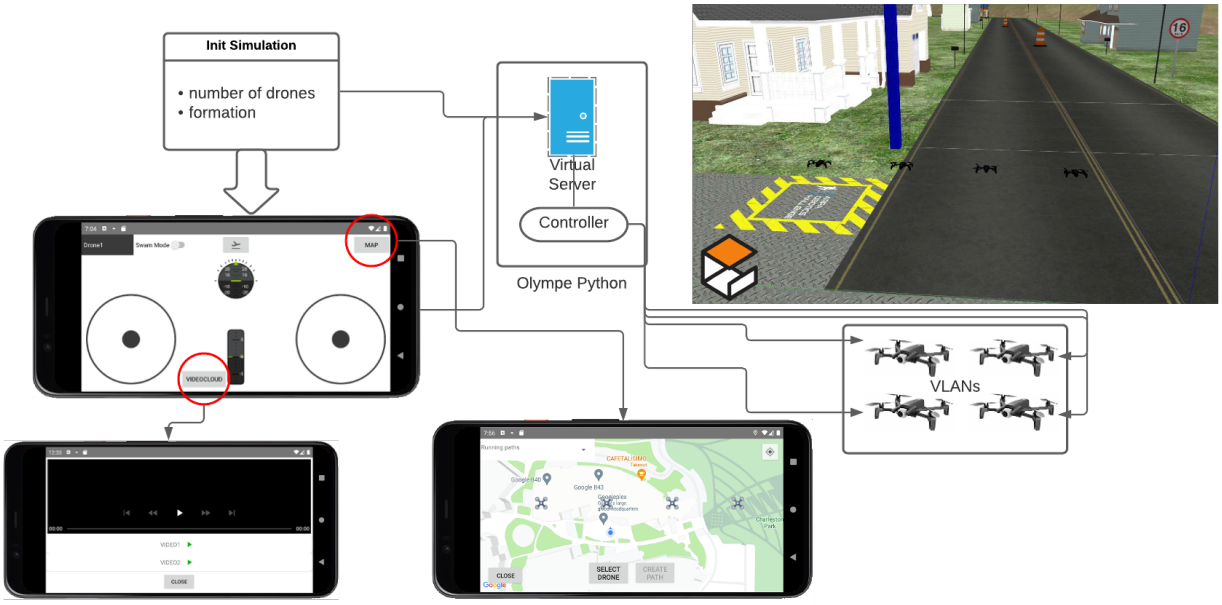

Figura 4.2: Schema multi-drone

l'operazione. Ha il pieno controllo dei velivoli essendo connesso a ognuno di essi. I percorsi pianificati possono essere intrapresi anche da gruppi di droni. Per esempio simulando quattro droni, due droni possono intraprendere lo stesso percorso, mentre i restanti essere a disposizione dell'utente per altre operazioni.

4.3 Studio di sistemi di controllo di swarm

Avendo a disposizione questo framework, abbiamo pensato di svolgere qualche esperimento per il controllo di uno swarm e analizzarne il comportamento utilizzando un algoritmo costruito appositamente per il caso. Nei paragrafi seguenti verranno presentati i vari studi fatti.

4.3.1 Algoritmo di controllo

L'algoritmo utilizzato per i nostri esperimenti è stato costruito sulla base di Virtual Spring Forces (VSP) presentato in questo articolo [32]. Lo scopo

principale di VSP è di mantenere un mesh di nodi a una distanza prefissata. Collegando ognuno di essi con i propri vicini tramite dei link che si comportano similimente a una "molla". Se un nodo si avvicina o si allontana rispetto a un altro, il link reagirà generando una forza calcolata utilizzando la legge di Hook. Per realizzarlo ogni drone avrà un proprio thread dedicato dove verrà eseguito l'algoritmo.

Funzionamento

L'algoritmo prende in input due valori: (i) stiffness per la rigidezza del link, (ii) distanza desiderata tra i droni. Ogni drone avrà il proprio thread dedicato per l'esecuzione.

A ogni ciclo:

1. Creazione della lista vicini utilizzando Check Acute Angle, una funzione che controlla, per ogni drone, se questo può essere inserito nella lista. Viene calcolato un angolo utilizzando come vertici il velivolo, il vicino candidato ed a turno ogni drone restante. Se uno degli angoli formatosi è maggiore di 90 gradi, la funzione si interrompe e ritorna falso, non farà parte della lista.

2. Lista delle forze, per ogni vicino viene calcolata la forza vettoriale tramite calculate force, funzione che utilizza la legge di Hook:

$$Fe = -kx$$

$$xris = xunit * stiffness * displacement$$

$$yris = yunit * stiffness * displacement$$

displacement è la differenza tra la distanza tra i due velivoli e la distanza desiderata.
xunit e *yunit* sono le due unità del vettore di spostamento calcolato in direzione del vicino.

3. Forza Totale, la somma di tutte le forze presenti nella lista creata precedentemente. Questo vettore finale, è lo spostamento che dovrà eseguire il drone per mantenere la sua posizione all'interno dello swarm.

4.3.2 Esperimenti

Oggetto dello studio è stato analizzare al variare dello stiffness, come evolve mediamente la distanza tra le coppie di droni vicini nel tempo, e la velocità mantenuta di ogni velivolo. Gli esperimenti che presento sono stati svolti utilizzando quattro droni simulati, posizionati in linea come in figura 4.2 a una distanza iniziale di due metri.

Primo esperimento stiffness 1.0

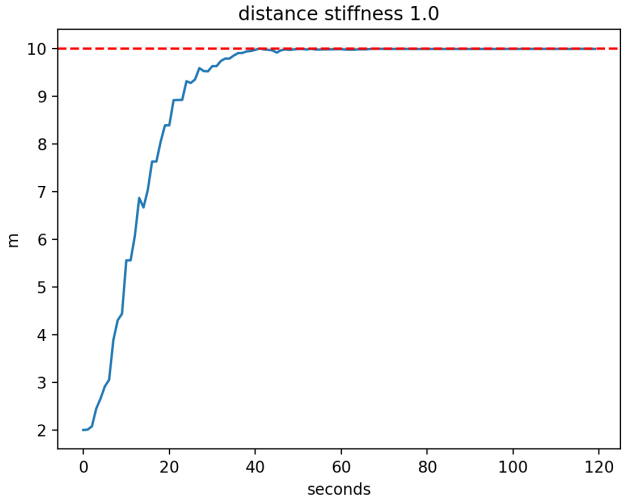

Figura 4.3: Distanza media tra le coppie di droni vicini

Dal grafico in figura 4.3 con valore di stiffness 1.0 possiamo osservare come nei primi 40 secondi la distanza da 2 metri cresce discretamente fino a raggiungere i 10 metri, la distanza desiderata. Da questo momento fino alla conclusione la distanza viene mantenuta stabilmente e lo swarm rimane

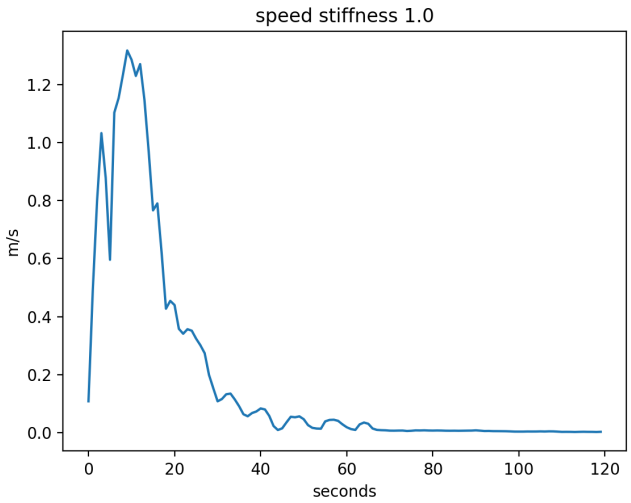

Figura 4.4: Velocità media dei droni

stabile. La velocità, rappresentata in figura 4.4, cresce rapidamente e raggiunge il suo picco dopo circa 17 secondi per poi descrescere quasi con la stessa rapidità e stabilizzarsi lentamentamente dal quarantesimo secondo.

Secondo esperimento stiffness 2.0

In figura 4.5 con stiffness raddoppiata a 2.0 la curva della distanza media raggiunge i 10 metri intorno al secondo 35, la stabilizzazione non è mai raggiunta, ma, al contrario, ad ogni intervallo di 10 secondi, cresce la differenza tra la distanza desiderata e l'effettiva. La curva della velocità, nella figura successiva 4.6, in concomitanza del raggiungimento dei 10 metri di distanza, è al minimo. A causa dell'instabilità dello swarm la curva risale oscillando progressivamente con picchi oltre i 2.5 m/s.

Terzo esperimento stiffness 4.0

Quest'ultimo esperimento al contrario dei precedenti ha una durata di 80 secondi e la stiffness è stata impostata a 4.0. Sul grafico della distanza 4.7 la curva tocca i 10 metri molto prima rispetto alle prove precedenti, meno

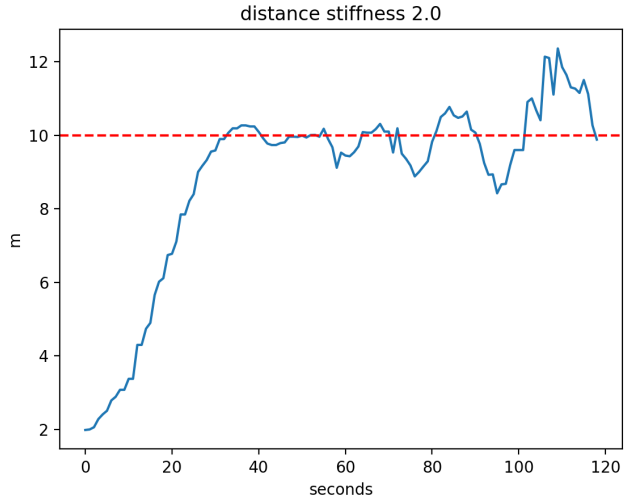

Figura 4.5: Distanza media tra le coppie di droni vicini

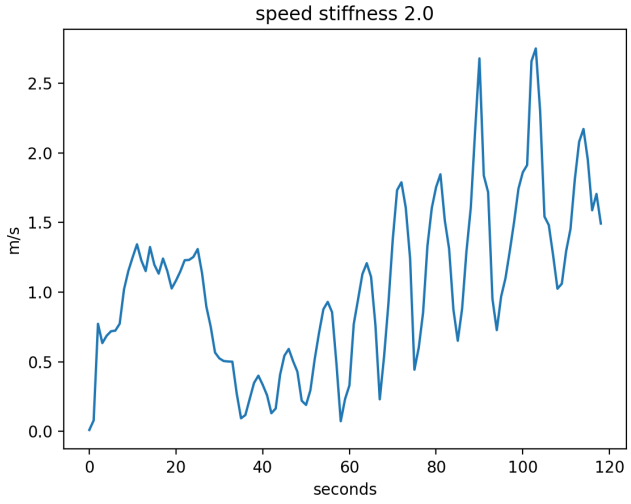

Figura 4.6: Velocità media dei droni

di 20 secondi, ma viene superata subito di 4 metri prima di oscillare. Dal ventesimo secondo in poi si può notare come l'ampiezza delle oscillazioni, intorno alla retta, diminuisce molto lentamente, accennando un tentativo di stabilizzazione. La velocità, nella figura successiva 4.8, non è mai stabile. Al secondo in cui vengono raggiunti i 10 metri, iniziano le oscillazioni, alcune

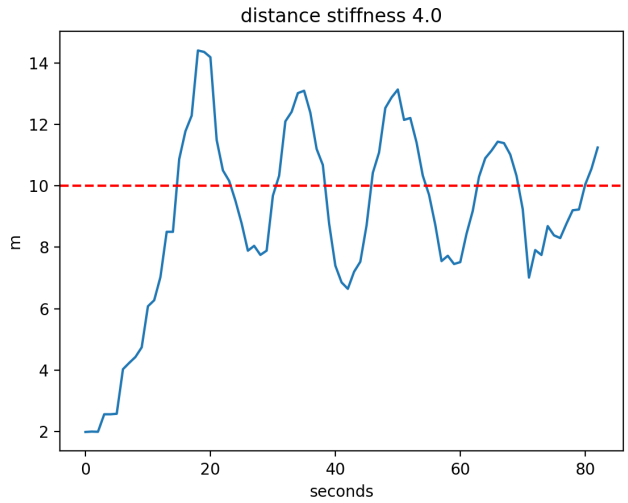

Figura 4.7: Distanza media tra le coppie di droni vicini

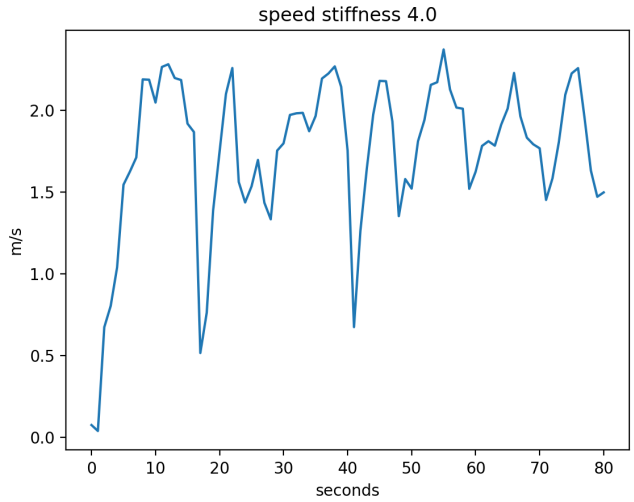

Figura 4.8: Velocità media dei droni

con ampiezza di 2.0 m/s. Dal secondo 50 l'ampiezza diminuisce della metà, sintomo della stabilizzazione percepita anche nel grafico della distanza.

Analisi

Dai tre esperimenti si evince che l'algoritmo è sensibile alle variazioni dello stiffness. Aumentadolo di una unità si è osservato come la stabilizzazione della curva intorno al valore 10 è fallita, e l'oscillazione aumentava progressivamente. Sul simulatore si poteva assistere al movimento circolare e oscillatorio dei droni per cercare vanamente di stabilizzarsi. Il raggio dell'oscillazione era sempre più elevato. Nell'ultimo esperimento lo stiffness è troppo grande per permettere una stabilizzazione ai primi tentativi. Infatti la retta viene passata immediatamente, causa la velocità elevata dei droni. Si osserva però come lentamente diminuisca l'estensione delle oscillazioni nel tentativo dello swarm di stabilizzarsi.

Conclusioni

In questo lavoro ho voluto proporre un framework per gestire e governare uno swarm di droni simulati. Questo sistema è in grado di pilotare uno o più velivoli utilizzando una applicazione Android sviluppata per lo scopo. L'utente può così scegliere quanti droni vuole simulare e ha a disposizione un ampio ventaglio di features; con il proprio smartphone sarà in grado di pilotarli, anche singolarmente, utilizzando i joystick, pianificare dei percorsi dalla mappa accessibile sull'app e visionare lo streaming video di ogni singolo velivolo. Tra le funzionalità sperimentali la modalità leader-follow, lo swarm segue un drone designato come capo pilotandolo manualmente. La simulazione dei droni avviene all'interno del simulatore Gazebo tramite Sphinx, un tool simulator creato dalla Parrot per la virtualizzazione dei propri velivoli. L'ambiente su Gazebo integra un motore fisico per dare la possibilità di eseguire simulazioni realisticamente avanzate, inserendo strade, palazzi, acqua e il vento. Potenzialmente il suo ambito di utilizzo può essere molteplice: sorveglianza delle colture, missioni di ricerca e salvataggio, ausilio nell'edilizia, consegna di pacchi ecc.... E' stato scelto come modello delle nostre simulazioni il drone Anafi Parrot, un drone commerciale e professionale con a bordo tutte le tecnologie più moderne, per avere la possibilità di realizzare le funzionalità desiderate. Nello sviluppo di un sistema per gestire uno swarm di droni è stato introdotto un algoritmo, con il quale sono stati realizzati degli esperimenti per osservare il comportamento dei velivoli e analizzarne i dati. E' stato possibile calibrarlo per riuscire a mantenere la stabilità e integrità dello stormo.

4.4 Lavori futuri

Il framework costruito è ampiamento aperto a miglioramenti, integrazioni e sviluppi. La possibilità tramite Olympe di potersi connettere anche a droni reali apre le porte al suo utilizzo fuori dagli ambieni simulati. Prima di questo occorre però migliorare il sistema e le sue features:

1. Realizzare un decoder per la codifica H.264 Main Profile dello streaming video sull'applicazione Android. Inoltre utilizzando le API di GroundSDK implementare la versione dello streaming per i droni reali, più facile da realizzare.

2. Perfezionare l'algoritmo di gestione dello swarm, utilizzando come funzione di movimento pcmd, generando così valori di pitch roll yaw e gaz a ogni ciclo al contrario di un vettore. In questo modo il movimento dei droni sarà più fluido e lineare, e il leader-follow sarà notevolmente migliorato.

3. Migliorare e ampliare il Path Following. Sull'applicazione Android la possibilità di mettere in pausa il percorso e riprenderlo, cambiare i droni in uso e scegliere l'ordine di esecuzione delle posizione scelte.

4. Aggiungere un gestore degli errori e problematiche derivanti dai droni sull'applicazione Android. Per esempio la batteria scarica.

5. Sperimentare il sistema con un numero di droni maggiore di 5.

6. Aggiungere tutte le features già disponibili sulle applicazioni Parrot, come lo scatto di fotogrammi, controllare il movimento della telecamera, Follow-Me e Cameraman per seguire rispettivamente il pilota o l'oggetto in movimento, modalità di pilotaggio come sport e race ecc. . . .

7. Implementazione di una modalità Multi-swarm. Si potranno creare più stormi selezionando i droni per formare i gruppi. Moltiplica le potenzialità e il multi-tasking rispetto a un solo swarm.

Parrot ha da poco annunciato l'uscita di un nuovo drone, AnafiAI [33], con tecnologie altamente all'avanguardia, come un modulo 4G LTE, rilevazione e aggiramento degli ostacoli tramite intelligenza artificiale, Autopilot e creazione di modelli 3D dell'ambiente scansionato con la videocamera sempre attraverso l'AI, ed altre nuove funzionalità. Sphinx e Olympe verranno aggiornati e migliorati appositamente per il nuovo modello, insieme al rilascio di un nuovo SDK per iOS e Android, una nuova applicazione smartphone open-source e l'utilizzo del simulatore Unreal Engine, più performante e realistico. Il mio framework potrebbe giovarne in termini di performance e stabilità della simulazione, come virtualizzare e pilotare uno swarm più numeroso, integrare il nuovo modello Anafi e sviluppare nuove funzionalità con le sue tecnologie.

Bibliografia

[1] R. Schneiderman, "Unmanned drones are flying high in the military/aerospace sector [special reports]," *IEEE Signal Processing Magazine*, vol. 29, no. 1, pp. 8–11, 2012.

[2] G. Horsman, "Unmanned aerial vehicles: A preliminary analysis of forensic challenges," *Digital Investigation*, vol. 16, pp. 1–11, 2016.

[3] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Search and rescue operation using uavs: A case study," *Expert Systems with Applications*, vol. 178, p. 114937, 2021.

[4] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2624–2661, 2016.

[5] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Uavs for smart cities: Opportunities and challenges," pp. 267–273, 05 2014.

[6] E. K. Elsayed, A. M. Alsayed, O. M. Salama, A. M. Alnour, and H. A. Mohammed, "Deep learning for covid-19 facemask detection using autonomous drone based on iot," in *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pp. 1–5, 2021.

[7] L. Williams, "Nature lessons: We are familiar with drones that look high-tech but rather robotic, sprouting multiple sets of rotors or with long imposing fixed wings. but the uavs of the future might look less like vehicles and more like insects, birds or even bats. here we look at three nature-inspired examples," *Engineering Technology*, vol. 15, no. 3, pp. 76–77, 2020.

[8] G. Chmaj and H. Selvaraj, "Distributed processing applications for uav/drones: A survey," in *Progress in Systems Engineering* (H. Selvaraj, D. Zydek, and G. Chmaj, eds.), (Cham), pp. 449–454, Springer International Publishing, 2015.

[9] DJI, "Sz dji technology co." `https://www.dji.com`.

[10] Yuneec, "Yuneec international." `https://us.yuneec.com`.

[11] Parrot, "Parrot sa." `https://www.parrot.com/`.

[12] DJI, "Dji agras t30." `https://www.dji.com/it/t30/specs`.

[13] Parrot, "Parrot anafi." `https://www.parrot.com/en/drones/anafi`.

[14] G. Pau, A. Bazzi, M. E. M. Campista, and A. Balador, "Towards 5g and beyond for the internet of uavs, vehicles, smartphones, sensors and smart objects," *Journal of Network and Computer Applications*, vol. 135, pp. 108–109, 2019.

[15] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles — a survey," *Journal of Industrial Information Integration*, vol. 16, p. 100106, 2019.

[16] DJI, "Dji simulator." `https://www.dji.com/it/simulator`.

[17] ZEPHYR, "Zephyr simulator." `https://zephyr-sim.com`.

[18] Realflight, "Realflight sim." `https://www.realflight.com/index.php`.

[19] Autodesk, "Autodesk 3d." `https://www.autodesk.it`.

[20] E. Soria, F. Schiano, and D. Floreano, "Swarmlab: a matlab drone swarm simulator," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8005–8011, 2020.

[21] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, 2004.

[22] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo," in *Simulation, Modeling, and Programming for Autonomous Robots* (I. Noda, N. Ando, D. Brugali, and J. J. Kuffner, eds.), (Berlin, Heidelberg), pp. 400–411, Springer Berlin Heidelberg, 2012.

[23] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6235–6240, 2015.

[24] O. Michel, "Cyberbotics ltd. webots™: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[25] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, 2013.

[26] O. R. e. a. Pinciroli C., Trianni V., "Argos: a modular, parallel, multi-engine simulator for multi-robot," *Swarm Intell*, vol. 6, p. 271–295, 2012.

[27] N. R. Zema, A. Trotta, E. Natalizio, M. Di Felice, and L. Bononi, "The cuscus simulator for distributed networked control systems: Architecture and use-cases," *Ad Hoc Networks*, vol. 68, pp. 33–47, 2018.

Advances in Wireless Communication and Networking for Cooperating Autonomous Systems.

[28] S. Shah, D. Dey, C. Lovett, and A. Kapoor, *AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles*, pp. 621–635. 01 2018.

[29] A. Banerjee and A. Roychoudhury, "Future of mobile software for smartphones and drones: Energy and performance," in *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pp. 1–12, 2017.

[30] M. M. Bachtiar, F. Ardilla, and A. A. Felinanda, "Android application design as ground control station (gcs) and waypoint navigation in unmanned aerial vehicle (uav)," in *2019 International Electronics Symposium (IES)*, pp. 299–306, 2019.

[31] Z. Lu, F. Nagata, and K. Watanabe, "Development of ios application handlers for quadrotor uav remote control and monitoring," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 513–518, 2017.

[32] K. Derr and M. Manic, "Extended virtual spring mesh (evsm): The distributed self-organizing mobile ad hoc network for area exploration," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 12, pp. 5424–5437, 2011.

[33] Parrot, "Parrot anafi ai photogrammetry." `https://www.parrot.com/en/drones/anafi-ai/technical-documentation/photogrammetry`.