
UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA TRIENNALE IN INFORMATICA

*Francesco Cristoforo Conti*


# Twinflie, un software per l'orchestrazione di uno sciame di droni

RELAZIONE PROGETTO FINALE


Relatore:
Prof. Federico Fausto Santoro
Prof. Corrado Santoro



Anno Accademico 2022 – 2023

# Indice

# Capitolo 1 - Introduzione

Gli Unmanned Aerial Vehicle (UAV), o droni, oggi giorno sono sempre più utilizzati in una varietà di applicazioni come sorveglianza [1], mappatura [1], operazioni di recupero a causa di disastri [2,3], l'acquisizione di informazioni spaziali, la raccolta di dati da aree inaccessibili, il monitoraggio del traffico [4], etc.

Il problema sussiste quando è necessario eseguire le suddette operazioni in luoghi sconosciuti e, per risparmiare sia denaro che tempo, risulta comodo avere la possibilità di testare le missioni che i droni dovranno effettuare in ambienti virtuali. Questo consente di eseguire operazioni senza il rischio di danneggiamento di oggetti o persone, risparmiando così delle risorse preziose accelerandone lo sviluppo ed il testing.

Inoltre, questo progetto si propone di essere adatto, non solo a simulare il comportamento di un drone, ma anche di uno sciame. A tale fine, diventa quindi fondamentale un sistema per orchestrare tale architettura di droni.

## 1.1 - UAV

Gli UAV, comunemente noti come droni, sono dei velivoli senza persone a bordo il cui volo può essere controllato sia in remoto da un operatore umano che in maniera autonoma.

Gli UAV vennero creati all'inizio degli anni 2000 per scopi militari, per quelle missioni reputate troppo pericolose per avere un equipaggio umano, e sono diventate un asset molto importante per le guerre più moderne. Negli ultimi anni, grazie all'abbassamento dei costi e al miglioramento delle tecnologie legate alla sensoristica digitale, si sono diffusi gli UAV anche a campi non militari e ricreativi. In particolare, hanno trovato un larghissimo uso in campo di sorveglianza, monitoraggio e ricognizione.

A livello strutturale i droni usati per scopi civili sono generalmente dei multirotori, come questo in figura 1.1.

**Figura 1.1 – Dji mavic mini un piccolo drone per uso civile**

Tipicamente i droni multirotore sono caratterizzati da: un insieme uniforme di eliche orizzontali, tipicamente di numero maggiore o uguale a 4, disposte in maniera simmetrica a forma circolare, una cellula simmetrica/bilanciata anche se non strettamente obbligatoria, la capacità di decollo orizzontale e di atterraggio, la possibilità di muoversi in 4 direzioni (XYZ e rotazionale), nessuna criticità dal punto di vista meccanico-aereodinamico e il controllo totale nel software senza nessuna parte meccanica di controllo.

Il sistema di riferimento tipico di un multirotore è quello in figura 1.2.

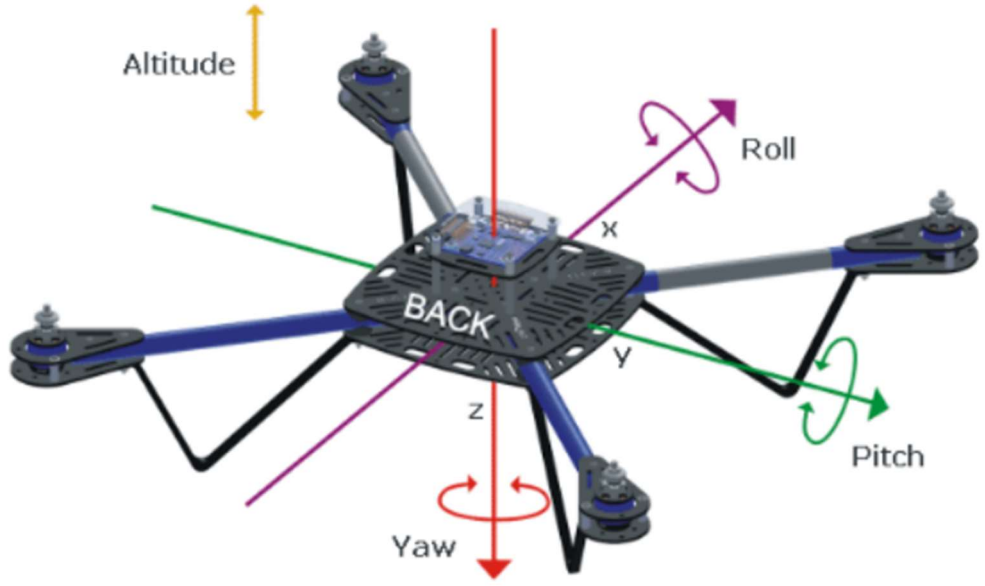


**Figura 1.2 – sistema di riferimento tipico di un multirotore**

La posizione di un multirotore nello spazio è quindi espressa dalla sestupla {X, Y, Z, θ, φ, ψ} dove φ è l'angolo di roll, θ è l'angolo di pitch e ψ l'angolo di yaw.

Il modello a 4 rotori disposti a croce è sicuramente il più diffuso e utilizzato, ma esistono altri possibili modelli con 3, 6 oppure 8 rotori come in figura 1.3.

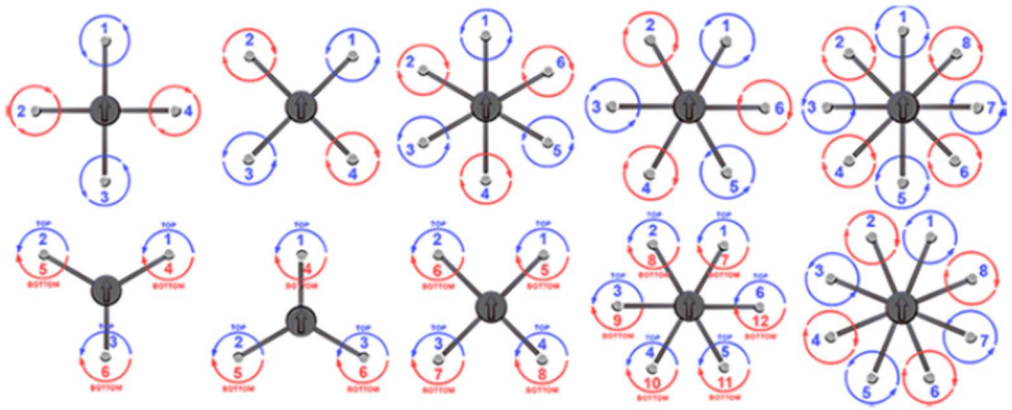

**Figura 1.3 –possibili modelli di multirotori**

Il funzionamento nei modelli con un numero pari di rotori è simile ai modelli con quattro motori. Le eliche, infatti, devono ruotare a coppie e in direzioni opposte per bilanciare la terza legge di newton, inoltre, le eliche devono avere passi opposti a coppie.

Ad esempio, nel caso di 4 motori disposi a X, quelli disposti sull'asse \ devono ruotare in un verso (orario o antiorario), invece, i rotori disposti sull'asse / devono ruotare nel verso opposto, come in figura 1.3 secondo modello. Lo scopo è quello di bilanciare il terzo principio della dinamica, infatti se i 4 rotori rotassero tutti nello stesso verso il drone ruoterebbe in maniera incontrollata su se stesso, mentre ruotando in direzioni opposte la forza totale che agisce sul corpo si equilibra riuscendo a volare in maniera stabile.

Lo stesso problema è presente in un modello con un numero di dispari di rotori. In questo caso è necessario che una delle tre eliche si muova in verso opposto rispetto alle altre due, per bilanciare la terza legge della dinamica; ma è anche necessario che la spinta aerodinamica dell'elica, che ruota in senso opposto, sia pari alla somma delle spinte delle altre due eliche. Inoltre, è necessario che la velocità di tale elica sia maggiore rispetto a quella delle altre due. Questi limiti complicano notevolmente la gestione delle eliche del drone e rendono il modello meno vantaggioso rispetto ai modelli con un numero pari di eliche.

Come è stato accennato per il modello a tre rotori, per effettuare un qualsiasi movimento è necessario modulare la velocità delle singole eliche in maniera opportuna, mentre nel modello a quattro rotori, il controller fa ruotare i rotori alla stessa velocità.

Ad esempio, ai fini del decollo, sarà necessaria aumentare la velocità dei rotori per ottenere una spinta aerodinamica maggiore della forza peso del drone, mentre abbassando leggermente la velocità dei rotori rispetto alla velocita di decollo il drone riesce a restare alla stessa altezza.

Aumentando, invece, la velocità dei rotori che ruotano in senso orario o antiorario si ottiene una rotazione della yaw del drone.

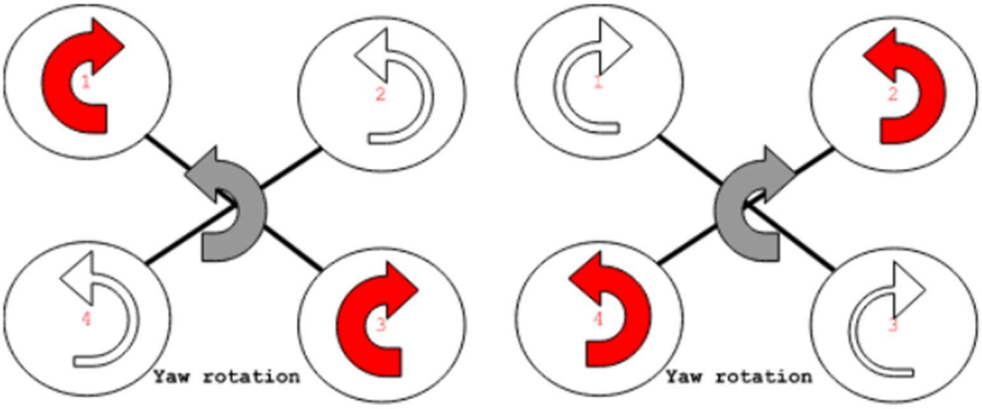


**Figura 1.4 – rotazione di yaw**

In maniera analoga è possibile ottenere una rotazione di pitch aumentando la velocità delle eliche anteriori e posteriore.

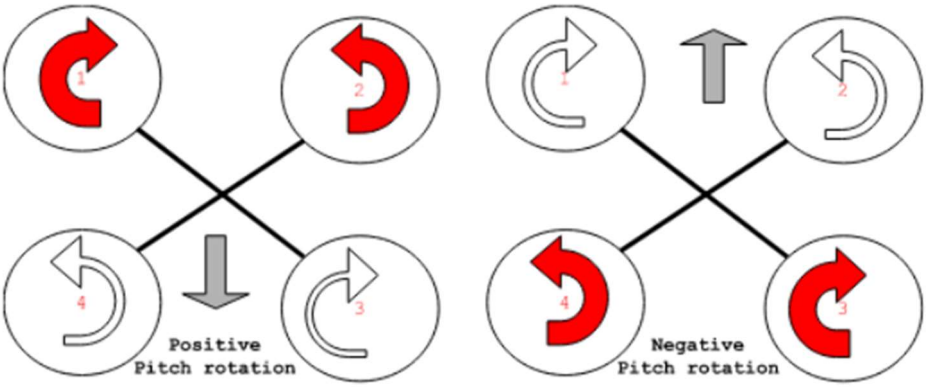


**Figura 1.5 – rotazione di pitch**

Allo stesso modo, aumentando la velocità delle eliche laterali è possibile ottenere una rotazione nel senso di roll come in figura 1.6.

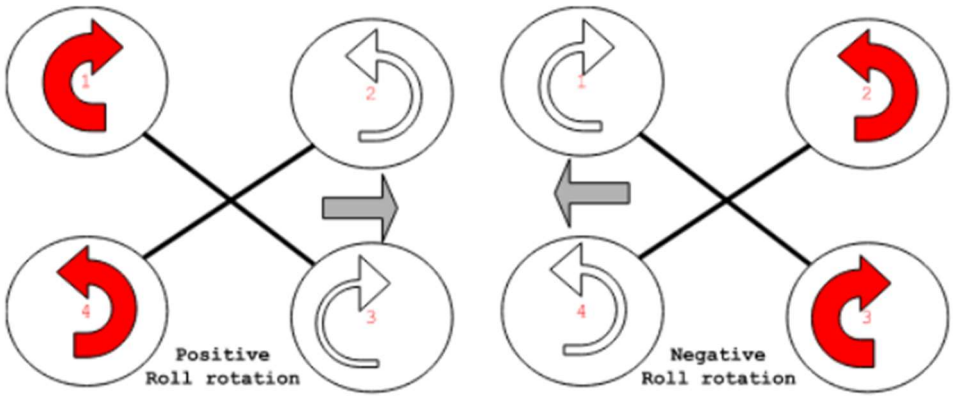


**Figura 1.6 – rotazione di roll**

## 1.2 - Digital Twin

Un gemello digitale [5] (digital twin) è un oggetto simulato su un computer che ha le stesse caratteristiche dell'oggetto reale. Grazie a questa tecnologia è possibile prevedere guasti e malfunzionamenti simulando le sollecitazioni a cui sarà sottoposto l'oggetto durante il suo utilizzo.

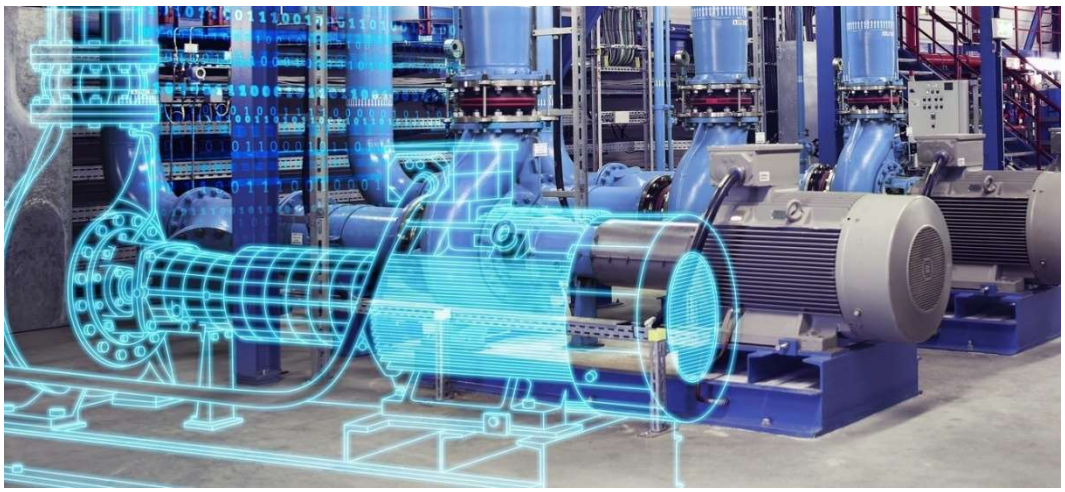

**Figura 1.7 – digital twin**

Per esempio, in campo industriale [6] è possibile creare una copia di un impianto con le stesse caratteristiche fisiche dei materiali usati, da questo punto di partenza è possibile simulare tutte le sollecitazioni a cui esso sarà sottoposto e quindi sarà possibile gestirlo in remoto per prevedere guasti e malfunzionamenti, ottimizzare la produttività e gestire le risorse.

Il concetto di digital twin è quindi sempre più attuale e trova innumerevoli applicazioni in diversi campi, come quello automobilistico dove è possibili fare simulazioni relative allo stato delle sue componenti.

Sarà quindi possibile prevedere la manutenzione straordinaria del veicolo garantendo uno stato ottimale della macchina e una migliore transizione allo stato ecologico.

### 1.3 - Sciami di droni

L'architettura di UAV fino ad ora descritta può essere ulteriormente migliorata utilizzando un insieme di droni anziché uno solo, questo permette una migliore tolleranza degli errori e una maggiore efficienza nelle operazioni. Ad esempio, si supponga di voler svolgere un compito di sorveglianza all'interno di un edificio: con un solo drone potremmo perlustrare poche sezioni alla volta; avendo a disposizione uno sciame di droni il compito risulterebbe estremamente più efficiente. In maniera simile, se il task da eseguire fosse di mappare un terreno di

diversi chilometri o di sorvegliare un bosco per prevenire incendi, un singolo drone potrebbe coprire una piccola area di terreno alla volta, mentre, una serie di droni completerebbe lo stesso task in maniera più efficiente, sia in relazione al tempo sia in relazione alla precisione. Sarebbe possibile avere una maggiore quantità di dati su cui applicare della ridondanza, coprire zone di terreno con più droni e garantire la copertura di angoli ciechi nella sorveglianza di edifici.

I lati negativi di avere più droni che operano sullo stesso task sono il costo di più dispositivi e la inter-coordinazione degli stessi. Il primo limite può essere bilanciato dal risparmio in termini di tempo e di efficienza che più droni permettono. Per il secondo è necessario sviluppare dei software appositi per la coordinazione dei droni.

Il progetto qui presentato si propone come soluzione di quest'ultima limitazione.

# Capitolo 2 - Stato dell'arte

Questo progetto si propone come un simulatore di sistemi robotici in ambiente virtuale in grado di coordinare le operazioni tra tali sistemi. Si fornisce di seguito una disamina delle principali soluzioni che permettono di effettuare delle simulazioni nello sviluppo di sistemi robotici.

## 2.1 - Ros e Gazebo


**Figura 2.1 – ros e gazebo**

Uno dei modi più utilizzati per simulare sistemi robotici è la combinazione di Ros [7] e Gazebo [9].

Ros (Robot Operating System) è un middleware per la robotica che fornisce un insieme di framework software per lo sviluppo di sistemi robotici, di natura differente, che implementa il controllo a basso

livello dei dispositivi, sensori, motori, etc., la comunicazione di messaggi tra processi e la gestione dei pacchetti.

Gazebo, invece è un simulatore robotico che implementa un motore fisico e il render 3d delle simulazioni.

Per permettere la simulazione dei droni senza un hardware fisico nell'architettura Ros Gazebo viene spesso usato SITL (software in the loop) [10] che permette di far eseguire il software dei droni direttamente sul pc senza avere l'hardware apposito.

Unendo queste due tecnologie è possibile: simulare un robot sfruttando il render grafico e fisico di Gazebo, programmare il software in Ros e simulare l'hardware con SITL.

Questa architettura consente di eseguire della simulazione ultra-realistiche del sistema che si sta sviluppando, allo stesso tempo però, si denota la difficoltà di gestione di tale architettura in quanto risulta computazionalmente complessa o onerosa. Inoltre, si sottolinea la profondità dell'architettura di tali sistemi in quanto Gazebo si interfaccia con Ros che a sua volta andrà ad interfacciarsi con SITL che, infine, comunicherà nuovamente con Gazebo rendendo così molto complesso gestire tale struttura.

In particolare, esiste un progetto basato sull'architettura descritta, specificamente per i crazyflie [8], che verranno usati per questo progetto.

La differenza sostanziale tra il progetto discusso e quello realizzato in Gazebo e Ros è che il primo permette di testare i percorsi di diversi droni contemporaneamente e di inviare un comando ai droni per eseguire questi percorsi nel mondo reale, il secondo, invece, permette una simulazione molto dettagliata del volo dei droni, a costo di un superiore costo computazionale che non permette la simulazione di diversi droni contemporaneamente e non permette, allo stato attuale, di avviare l'esecuzione di un volo testato nel mondo reale.

## 2.2 - Swarm coordinator

Esistono essenzialmente 2 categorie [14] di soluzioni per questo problema: rule-based e optimization-based. Le prime sono basate su un insieme di regole che i droni rispettano per volare in maniera coordinata, il secondo approccio invece è basato su algoritmi che preso in input l'obbiettivo e i limiti dei droni calcolano il percorso ottimale per lo sciame.

Il progetto in esame si propone come un incontro tra le due soluzioni.

Sarà possibile assegnare ad ogni drone un percorso da seguire e delle regole da rispettare. Inoltre, sarà possibile eseguire virtualmente tale percorso, controllare il comportamento di ogni dispositivo e monitorare l'esecuzione complessiva.

Come detto pocanzi non esiste, dalle ricerche effettuate, un simulatore pubblico che permetta di effettuare un lavoro del genere.

# Capitolo 3 - Tecnologie utilizzate

Si proceda ora ad una disamina di tutte le tecnologie necessarie alla costruzione del simulatore.

## 3.1 - Crazyflie

**Figura 3.1 – Crazyflie**

I droni usati per il progetto in esame sono dei CrazyFlie 2.1 [12], mostrati in figura 3.1. Essi sono dei droni dal peso di 27g con diversi moduli di espansione, pensati per rendere i droni più piccoli e versatili. Sono forniti inoltre di un software open source pensato per essere altamente e facilmente personalizzabile.

Per comunicare i droni utilizzano le onde radio, in particolare, utilizzano un'antenna radio per comunicare, in ricezione e trasmissione, con un computer.

Come sensoristica a bordo si trova: un giroscopio e un accelerometro. Entrambi producono dati soggetti a errore di varia natura, tali dati verranno usati dal filtro di Kalman per ridurre l'errore totale del sistema.

## 3.2 - Locoposition system

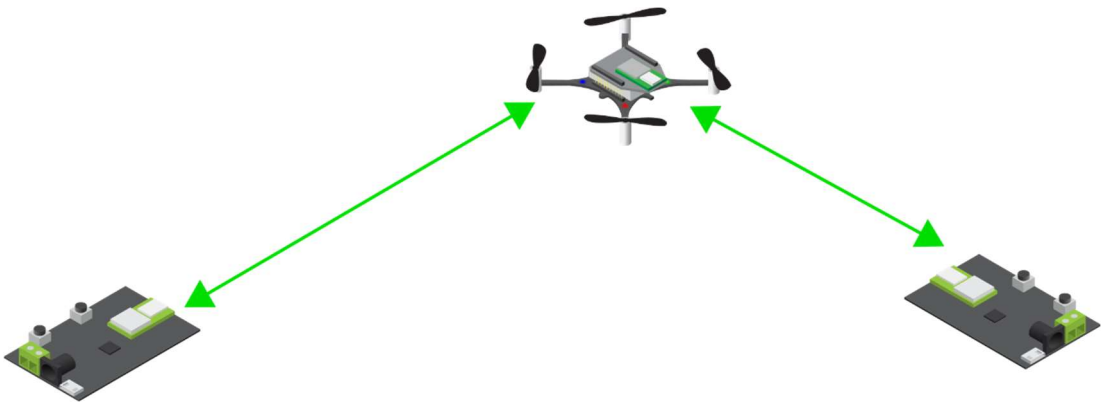

**Figura 3.2 – Loco Position system schema**

Per la localizzazione dei droni viene utilizzato il sistema Locoposition, il cui schema di funzionamento è illustrato in figura 3.2.

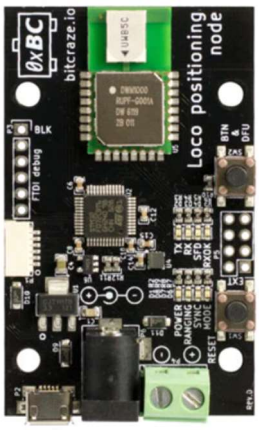


**Figura 3.2 – Loco Position node**

Esso consiste di una serie di ancore, ovvero i nodi in figura 3.3, con cui il drone comunica grazie ad un apposito modulo di espansione chiamato Loco Position deck.

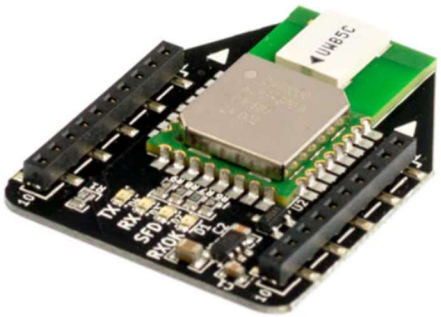

**Figura 3.3 – Loco Position deck**

Nel caso preso in esame sono state utilizzate otto ancore posizionate agli angoli di una stanza grazie alle quali il drone riesce a triangolare la propria posizione nello spazio.

## 3.3 - Python

**Figura 3.4 – Python logo**

Per comunicare con i droni è stato realizzato un server Python che espone delle chiamate api restless tramite la libreria Flask, e comunica con i droni tramite un wrapper specifico per i droni disponibile sul sito del produttore degli stessi.

Questa libreria è stata utilizzata principalmente per prendere la posizione stimata del drone, settare i valori dello stimatore di Kalman e inviare al drone le informazioni relative al movimento.

### 3.3.1 - Flask


**Figura 3.5 – Flask logo**

Flask è un micro-framework per il web scritto in Python che permette, lato server, una rapida creazione e gestione di api web e piccole pagine. Si definisce micro-framework in quanto non implementa alcuno strumento avanzato di astrazione.

Non essendo necessaria una robusta infrastruttura web è stato possibile utilizzare flask che a costo di una peggiore efficienza computazionale fornisce una maggiore rapidità e flessibilità nello sviluppo.

## 3.4 - Godot


**Figura 3.6 – Godot logo**

Godot è un motore per videogiochi estremamente leggero, grazie al quale è possibile realizzare programmi per una moltitudine di piattaforme, tra cui: dispositivi mobili come Android e IOS, dispositivi desktop quali linux, windows e macOS, web platform, e dispostivi per la realtà aumentata come Oculus Rift.

Godot supporta diversi linguaggi di programmazione, tra cui GDscript, C#, C++ e GDnative.

GDnative consente di programmare nativamente in diversi linguaggi di programmazione, tra cui c, e grazie al supporto della community sono supportati anche Rust, Nim, JavaScript, Haskell e altri. Per gli scopi di questo progetto è stato scelto GDscript che è un linguaggio o di scripting ad alto livello dalla sintassi simile a Python altamente ottimizzato per l'architettura a scene di Godot.

Godot è stato utilizzato per ottenere un ambiente virtuale real-time con cui è possibile interagire e gestire i droni. Questo ambiente utilizzerà le API messe a disposizione dal server realizzato in python.

Si preferisce utilizzare Godot ad altri motori fisici, come Gazebo o Unity, per la sua leggerezza e facilità nello sviluppo a monte di completezza nella simulazione della fisica, e per la possibilità di utilizzare il software sviluppato su una moltitudine di dispositivi, il che rende questo progetto, lato client, eseguibile su un qualsiasi dispositivo.

# Capitolo 4 - Twinflie

Si procede ad una accurata analisi della struttura interna del progetto.

## 4.1 - Comunicazione drone-ancore

La comunicazione tra le ancore e il drone avviene tramite onde radio per mezzo di uno specifico tag di localizzazione montato sul drone, questo sistema è ideale per localizzare il drone in ambienti interni.

Il sistema implementa nativamente 3 modalità di utilizzo: Two Way Ranging (TWR), Time Difference of Arrival 2 (TDoA 2) e Time Difference of Arrival 3 (TDoA 3).

Nella prima modalità, il tag emette un segnale in sequenza a tutte le ancore e grazie alla misurazione del tempo di risposta calcola la distanza da ogni ancora e determina così la propria posizione nello spazio. Questo sistema prevede un minimo di 4 ancore per determinare la posizione ma, sul sito del produttore [13], si consiglia di utilizzare 6 ancore per aggiungere ridondanza e accuratezza.

Nella seconda, invece, il sistema di ancore continua a inviare pacchetti per la sincronizzazione, con uno schema di invio dei pacchetti a slot temporali. Un drone può calcolare la differenza di tempo intercorsa tra l'arrivo di due pacchetti per stimare la distanza relative alle ancore che

avevano inviato i pacchetti, l'unica limitazione di questo sistema è che a causa degli slot temporali può lavorare al massimo con otto ancore.

Infine, TDoA 3 lavora in maniera analoga al precedente solo che anziché usare uno schema a slot temporali, utilizza uno scheduling dei pacchetti randomizzato, quindi, non vi è più la limitazione di otto ancore poiché il sistema può lavorare con illimitate ancore virtuali. Aumentando il numero di ancore è possibile andare ad espandere l'area di localizzazione su più stanze anche se esse non sono visibili tra di loro.

La seconda e la terza modalità, al contrario della prima, prevedono che siano le ancore a inviare pacchetti di sincronizzazione avendo così il vantaggio di non essere limitati all'utilizzo di un solo drone per volta. Inoltre, la terza modalità, per via del suo schema randomizzata, permette una gestione dinamica della connessione e della disconnessione delle ancore.

Si precisa che per la modalità TWR si ha una precisione di circa 10 cm, anche nel caso in cui il drone esca dall'area descritta dalle ancore. TDoA 2 e 3 hanno un'accuratezza simile ma solo nel caso in cui il drone si trovi esattamente all'interno dell'area descritta dalle ancore, perdendo precisione quando essi si trovano fuori dall'area di azione.

Ai fini della realizzazione di tale progetto si sceglie di utilizzare la terza modalità (TdoA 3) escludendo la prima e la seconda modalità. Nello specifico, la prima non permette di gestire uno sciame di droni mentre,

la seconda, non permette la gestione dinamica delle connessioni delle ancore.

### 4.1.1 - Filtro di Kalman

Come è stato detto il sistema di localizzazione descritto è affetto da errori di stima. Per mitigarli il modello Crazyflie implementa il filtro di Kalman.

Il filtro di Kalman è un efficiente filtro ricorsivo che valuta lo stato di un sistema dinamico sulla base di una serie di misurazioni soggette a rumore. Per le sue caratteristiche uniche, è un ottimo filtro per dei dati soggetti a errori e disturbi, agenti su sistemi gaussiani a media nulla.

Esso implementa in maniera facile la Sensor Fusion, ovvero l'unione dei dati proveniente da sensori differenti che permette una stima più accurata dei dati a partire da dati non sicuri provenienti da fonti diverse.

Nello specifico, il filtro di Kalman, nel caso in esame, utilizza i dati misurati nel tempo di cui conosce il rumore statistico, per produrre una stima, della posizione, che mira ad essere più accurata rispetto alle misurazioni singole.

Lavorando in maniera ricorsiva, il filtro calcola la media pesata rispetto allo stato precedente, calcolato dal filtro stesso, e alle nuove rilevazioni effettuate dai sensori, pesate a loro volta rispetto al rumore statistico degli strumenti utilizzati.

La ragione dell'utilizzo di una media pesata è che valori con minore incertezza avranno peso maggiore e valori con maggiore incertezza avranno peso maggiore, garantendo maggiore affidabilità.

Il risultato di questa media pesata sarà una nuova stima che si pone tra i dati misurati dai sensori nel mondo reale e la stima predetta dal modello nel tempo precedente. Tale stima avrà una incertezza inferiore rispetto alle singole misurazioni.

## 4.2 - Comunicazione drone-computer

Per far comunicare il drone con il pc viene usata un'antenna che comunica tramite onde radio col drone e che può essere controllata tramite API Python. Per la gestione e comunicazioni con i droni è disponibile una libreria pubblica, sviluppata dalla casa produttrice dei droni, che permette di comunicare con essi. A partire da questa libreria è stato costruito un wrapper che permette una migliore gestione dei drone per gli scopi di questo progetto rispetto alla libreria ufficiale.

Grazie a questo wrapper è possibile: ottenere la pozione stimata del drone grazie allo stimatore di Kalman in tempo reale, ottenere la posizione delle ancore, inviare al drone dei comandi relativi al movimento, gestire la connessione e la disconnessione del drone dal pc.

Per controllare gli spostamenti del drone nello spazio è stata realizzata una funzione che dato un punto nello spazio rappresentato dalle

coordinate (x, y, z), calcola di quanto è necessario variare la posizione del drone rispetto agli assi e successivamente invia un segnale al drone che gestirà tale spostamento. Inoltre, a un punto nello spazio può essere associato un comportamento particolare come, ad esempio, stare fermo per un determinato lasso di tempo, ruotare su stesso, o aspettare che uno o più droni siano arrivati a determinati punti identificati con un nome (meeting point) per poi eseguire un'azione.

Per implementare la gestione dei percorsi dei droni è stata realizzata una classe apposita che si comporta come una lista di punti, dove ogni punto è un dizionario Python che contiene la posizione del punto e le eventuali informazioni sul comportamento da eseguire.

Questa classe mette a disposizione due metodi: pop e passed. Il metodo pop, una volta invocato, ritornerà le coordinate del prossimo punto verso il cui il drone dovrà muoversi; il secondo, invece, deve essere invocato una volta che il drone ha superato l'ultimo punto inviatogli e gestirà un eventuale comportamento del drone in quel punto. Il costruttore di questa classe funziona nel seguente modo: il metodo riceve come parametri l'uri del drone, una struttura che contiene le istanze di tutti i droni attualmente connessi, una istanza della classe eventemitter e un json che contiene tutto il percorso che il drone dovrà seguire. Il costruttore effettuerà quindi il parsing del json e creerà ogni nodo della lista. Una volta creato questo oggetto, per ogni punto, verranno estratte le coordinate dello stesso e verrà inviato il segnale al drone di muoversi tramite l'apposita funzione prima descritta. Una volta che il drone è arrivato a destinazione viene

invocata la funzione passed che eseguirà uno dei comportamenti descritti in precedenza.

### 4.2.1 - Struttura del json dei percorsi dei droni

L'oggetto json che contiene tutti i percorsi dei droni è formato da un campo per ogni drone connesso. La chiave rappresenta il nome del drone e il corrispettivo valore è una lista di punti che indica il percorso che il drone dovrà eseguire.

Ogni punto della lista a sua volta è composto da un ulteriore oggetto json composto dai seguenti campi:

1. Name: il nome del punto, rappresentato da una stringa univoca.
2. Coordinate: le coordinate del punto.
3. Type: tipo di punto, utile a identificare un eventuale comportamento particolare del punto.
4. Meeting_name: se il punto è di tipo meeting_point sarà pari al nome del punto con cui il drone si dovrà sincronizzare.
5. Pause_time: se il punto è di tipo waiting sarà pari al tempo che il drone dovrà attendere fermo.

Un esempio di tale json è il seguente:

```
{"radio://0/90/2M/E7E7E7E702":[
    {"name":"punto_0","coordinate":"(0.32, -0.41, 1)","type":"base","meeting_name":"","pause_time":null},
    {"name":"punto_1","coordinate":"(1.17, -0.42, 0.97)","type":"base","meeting_name":"","pause_time":null},
    {"name":"punto_2","coordinate":"(1.17, 0.58, 0.97)","type":"takeoff","meeting_name":"","pause_time":null},
    {"name":"punto_3","coordinate":"(1.57, 0.58, 0.97)","type":"base","meeting_name":"","pause_time":null},
    {"name":"punto_4","coordinate":"(1.53, 0.58, -0.57)","type":"base","meeting_name":"","pause_time":null},
    {"name":"punto_5","coordinate":"(-0.73, 0.58, -0.59)","type":"base","meeting_name":"","pause_time":null},
    {"name":"punto_6","coordinate":"(-0.68, 0.58, 0.96)","type":"base","meeting_name":"","pause_time":null},
    {"name":"punto_7","coordinate":"(-0.68, -0.64, 0.96)","type":"landing","meeting_name":"","pause_time":null}]}
```


Nel Progetto qui presentato sarà compito di Godot generare questo json che verrà utilizzato dal server Python realizzato con Flask.

## 4.3 - Server Python

I dati ottenuti grazie alla comunicazione tra il drone e il pc vengono resi disponibili all'esterno del server Python grazie a delle API restfull realizzate tramite il modulo Flask.

Queste API, in particolare, permettono di connettersi a un drone specificandone l'uri radio, prendere la posizione delle ancore (azione possibile solo se almeno un drone è già connesso con il server), passare a ogni drone connesso un percorso e mandare il segnale a tutti i droni di avviare il percorso appena sincronizzato.

Il server implementa, inoltre, un protocollo per la distribuzione delle informazioni relative alle posizioni dei droni. Questo microservizio funziona nel seguente modo: un qualunque client o servizio può inviare un pacchetto UDP alla porta 20003 contenente la stringa “subscribe”, alla ricezione di tale stringa il server, tramite un thread

dedicato, salva l'indirizzo e la porta del mittente. Successivamente, tramite un altro thread, il server invierà a tutti i client sottoscritti un pacchetto che contiene le informazioni relative alle posizioni di tutti i droni connessi al server. Ogni client può, inoltre, disiscriversi dal servizio mandando la stringa "stop". In tal caso il server rimuoverà il client mittente dalla struttura che memorizza i client connessi, smettendo così di inviargli i pacchetti.

## 4.4 - Godot Client

Il client proposto è stato realizzato in Godot e ha lo scopo di mostrare all'utente finale la posizione del drone in un ambiente virtuale.

Il client proposto utilizza le chiamate API messe a disposizione dal server per ottenere le informazioni relative al drone e crearne un clone digitale.

Di seguito verranno spiegati e discussi gli oggetti che vengono utilizzati dal Client per gestire la scena.

### 4.4.1 - TouchObject

L'oggetto più importante di tutto il client è sicuramente il TouchObject, ovvero la rappresentazione dei droni nell'ambiente simulato.

L'albero della scena dei TouchObject è descritto in figura 4.1:

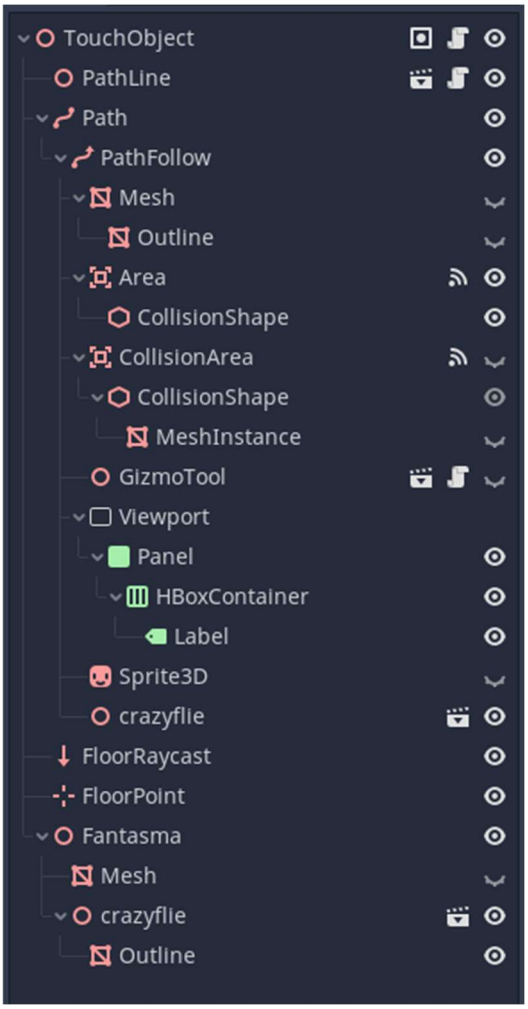


**Figura 4.1 – struttura TouchObject**

L'oggetto è composto da tre nodi principali: path, fantasma e pathline. Il primo è il drone simulato, utilizzato per l'interazione con l'utente e per testare i percorsi dei droni al fine di evitare le collisioni. Il secondo nodo, detto "fantasma", corrisponde al drone nella realtà, la cui posizione è aggiornata tramite il protocollo discusso nel paragrafo precedente. Infine, il terzo, contiene il percorso che i droni simulati dovranno testare e che potrà essere poi inviato ed eseguito dai droni nella realtà.

Il codice dell'oggetto touchobject gestisce: il movimento del drone simulato, l'aggiornamento della posizione del fantasma rispetto ai dati che arrivano dal server e la gestione delle collisioni tra i droni simulati.

Il risultato finale è visionabile in figura 4.2.

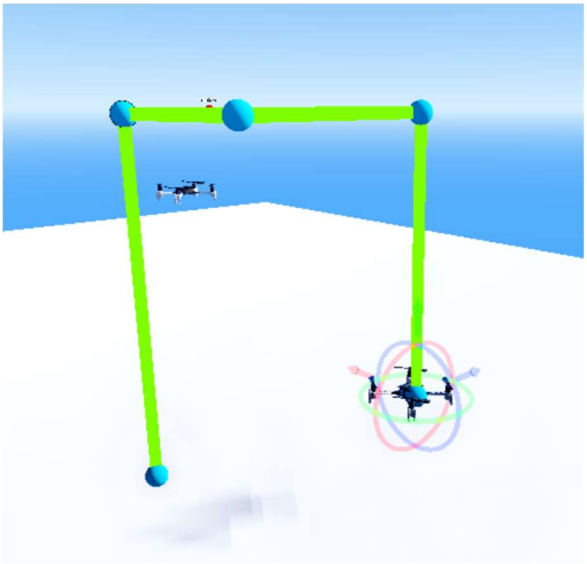

**Figura 4.2 – scena di droni**

Nell'immagine, il drone avvolto dalla gizmo è quello simulato, quello lievemente più indietro è il fantasma del drone reale, e la linea verde è il percorso contenuto nel nodo pathline che il drone dovrà eseguire.

### 4.4.2 - GUI

Le GUI sono la principale fonte di interazione con l'ambiente simulato. Sono state realizzate diverse implementazione interamente in Godot.

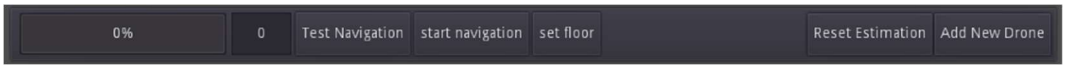


**Figura 4.3 – Bottom gui**

La prima con cui l'utente viene in contatto è posta nella parte bassa della schermata, figura 4.3, e contiene in ordine: la barra di progresso della simulazione, un bottone che permette di testare i percorsi impostati per i droni al fine di verificare eventuali collisioni, un bottone che permette di far partire la simulazione nel mondo reale, un bottone che permette di settare il livello 0 del pavimento della simulazione, un bottone che permette di resettare l'estimatore di kalman di tutti i droni e infine un bottone che permette di aggiungere nuovi droni alla scena.

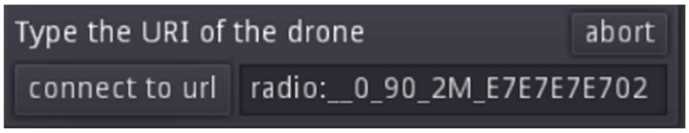


**Figura 4.4 – Connect window**

L'ultimo bottone in particolare apre la finestra presenta in figura 4.4 che permette di inserire l'uri del drone al quale ci si vuole connettere oppure di annullare l'operazione di connessione tramite il tasto abort.

Cliccando su un drone presente nella simulazione si apre la GUI presentata in figura 4.5.


**Figura 4.5 – Drone Gui**

Essa si compone di: nome del drone corrispondente che sarà la sua uri, un bottone che permette di aggiungere nuovi punti al percorso del drone, un bottone che permette di cancellare il drone dalla scena, un bottone che permette di deselezionarlo chiudendo di conseguenza questa finestra, un bottone che permette di sincronizzare la posizione del drone simulato rispetto alla posizione del drone reale e un bottone che permette di regolare la velocita, in m/s, della simulazione del drone.

Nel caso in cui venga premuto il bottone “add points” viene cambiata la prospettiva della scena a favore di una visuale dall’alto e si apre la gui in figura 4.6.

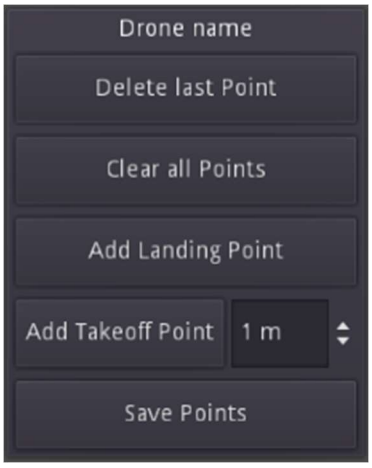


**Figura 4.6 – Adding point Gui**

Da questa scena è possibile aggiungere un nuovo punto cliccando con il puntatore sullo schermo, il nuovo punto verrà posizionato alle coordinate del puntatore del mouse con la stessa altezza del punto inserito precedentemente.

Analizzando la GUI si trovano: il nome del drone che è sempre l'uri del drone selezionato di cui stiamo aggiungendo punti al percorso, un bottone che permette di cancellare l'ultimo punto aggiunto al percorso, un bottone che permette di cancellare tutti i punti del percorso fino ad ora creato, un bottone che permette di aggiungere il punto di atterraggio che verrà posizionato sul pavimento perpendicolarmente rispetto all'ultimo punto inserito, un bottone che permette di aggiungere il punto di decollo che sarà posizionato sopra l'ultimo punto inserito ad una distanza corrispondente a quella specificata nella box a lato del bottone ed infine un bottone che permette di salvare e chiudere questa scena in cui è possibile aggiungere i punti.

Una volta salvati tutti i punti del percorso è possibile interagire con essi cliccando prima su un drone per evidenziare il percorso di quest'ultimo e poi cliccando su ogni punto del percorso che si desidera modificare, interagendo con un punto si aprirà la gui dedicata ad esso.


**Figura 4.6 – Point Gui**

Da questa interfaccia è possibile modificare le coordinate del punto selezionato cambiando il valore nel box della corrispettiva coordinata, inoltre, è possibile dare al punto dei comportamenti speciali selezionando le checkbox corrispondenti. Cliccando su una checkbox il punto selezionato assumerà quel comportamento, le altre checkbox verranno disabilitate e, se presenti, le informazioni aggiuntive relative comportamento al punto verranno mostrate per essere modificate.

Le tre possibili gui sono presenti in figura 4.8.

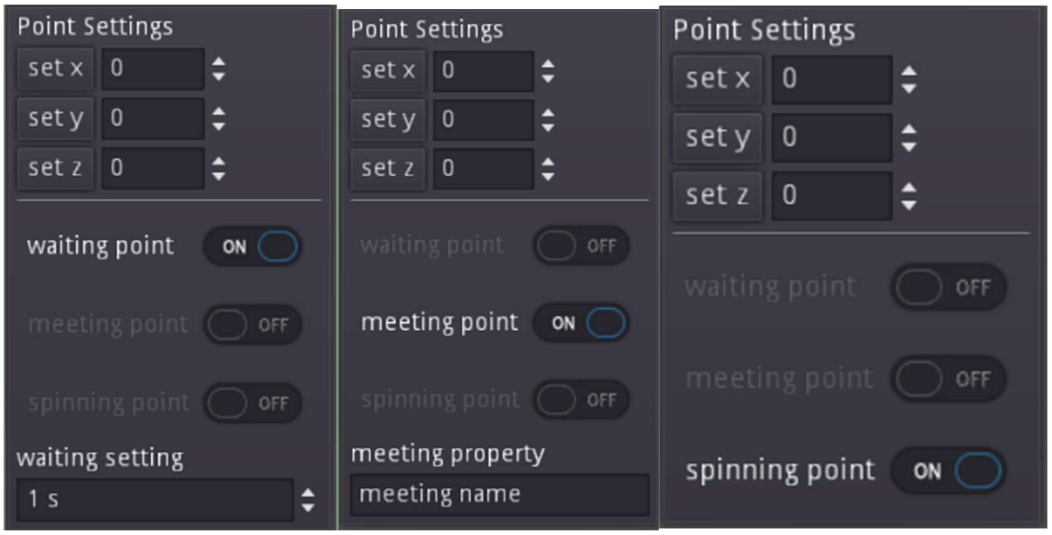


**Figura 4.8 – differenti Point Gui**

### 4.4.3 - Manager

I manager sono un insieme di oggetti automaticamente caricati all'interno della scena di Godot, che gestiscono a livello globale l'interazione tra oggetti e la creazione di oggetti.

Questi oggetti sono: SelectionManager, DroneManager, GuiManager e SceneManager.

Il SelectionManager gestisce l'oggetto attualmente selezionato, sia esso un Touchobject o un punto. In determinati momenti una funzione potrebbe aver bisogno di interagire con l'oggetto attualmente selezionato, con una apposita funziona dell'oggetto SelectionManager qualunque funzione avrà accesso all'oggetto attualmente selezionato.

Il secondo, il DroneManager, gestisce la creazione degli oggetti Touchobject.

Il terzo, il GuiManager, gestisce l'apertura e la chiusura delle varie GUI presenti nella scena.

Infine, lo SceneManager, gestisce la scena corrente che può essere di due tipi: addingstate e selectionstate. La prima è una vista dall'alto tramite cui è possibile interagire con la scena per aggiungere nuovi punti al percorso dei droni, utilizzando questa vista non è possibile muoversi nell'ambiente o modificare la posizione della camera. La seconda, è la vista di base in cui è possibile muoversi liberamente nell'ambiente e interagire con gli oggetti creati.

Questi manager descritti comunicano tra loro e con gli altri oggetti presenti nella scena per realizzare vari comportamenti. Per esempio, poniamo caso che a un certo momento l'utente interagisca con un drone. Questa interazione viene notificata, tramite un segnale, al SelectionManager che gestisce l'evento memorizzando l'oggetto selezionato in una variabile e notificando al GuiManager di aprire la gui del drone. Questa gui, a sua volta, prende i dati dell'oggetto selezionato e mostra a schermo le informazioni relative al drone. In maniera analoga è stata implementata la selezione di un punto.

# Capitolo 5 - Caso D'uso

Si analizza di seguito un esempio d'utilizzo dell'applicazione con due droni. Nello specifico verrà trattato: come installare l'applicativo, come inizializzarlo, come connettere i droni, come impostare i percorsi dei droni e come avviare il movimento nel mondo reale. Si definirà un percorso che ha la forma di un quadrato in cui ogni drone arrivato a un vertice aspetterà che l'altro arrivi al vertice opposto.

## 5.1 - Installazione e setup

Per prima cosa è necessario clonare il repository Git Hub del progetto. Quando la procedura sarà completata, essa avrà scaricato sul disco locale due cartelle: Godot client e Python server.

Per installare ed eseguire il client Godot non serve alcun prerequisito e basterà avviare l'applicativo presente nella cartella.

Per il setup del server è necessario aver installato Python e seguire la guida relativa all'utilizzo della libreria Crazyflie sul sito del produttore [16]. All'interno del progetto, nella cartella Python server, è già compresa la cartella crazyflie-lib-python contenente la libreria crazyflie, quindi è possibile saltare il passaggio "clonazione della repo" presente nella guida [16]. Si proceda effettuando il comando "pip install -e ." dentro la cartella crazyflie-lib-python che installerà la

libreria direttamente da questo progetto. Infine, sarà necessario eseguire il comando “pip install wsgiserver” per installare delle funzionalità aggiuntive di flask.

Una volta installato tutto il necessario, per avviare il server basterà avviare il file “main.py” contenuto dentro la cartella “test classe”.

## 5.2 - Descrizione dell’ambiente

Il test che si andrà ad analizzare è stato effettuato nel laboratorio di robotica del dipartimento di matematica e informatica dell’università di Catania. Verranno utilizzate otto ancore alimentate tramite alimentazione esterna, disposte negli angoli della stanza, poste su dei supporti di legno come in figura 5.1.


**Figura 5.1 – supporto delle ancore**

I droni, invece, sono stati disposti su un tavolo come in figura 5.2.

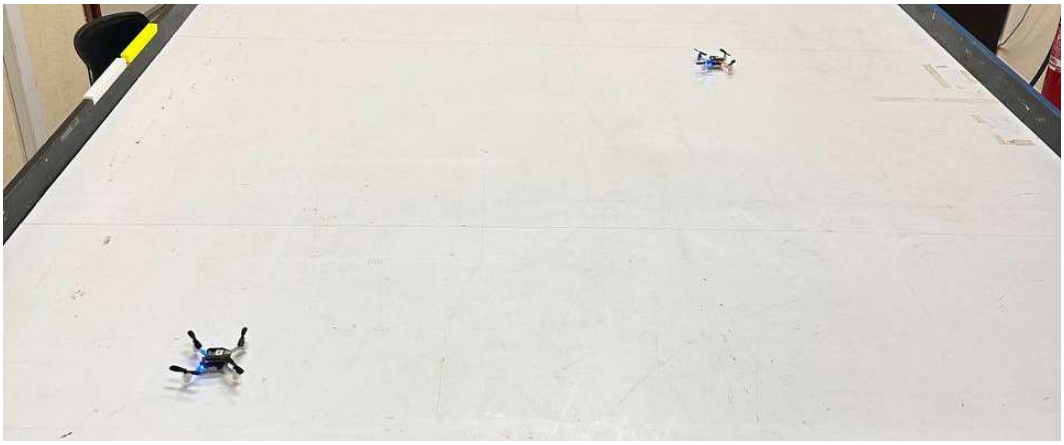

**Figura 5.2 – disposizione iniziale dei droni**

## 5.3 - Avvio dell'applicativo

Si procede avviando l'applicativo Godot e il server in Python e si aprirà la schermata in figura 5.3. Una volta avviati entrambi si può procedere alla connessione dei droni.

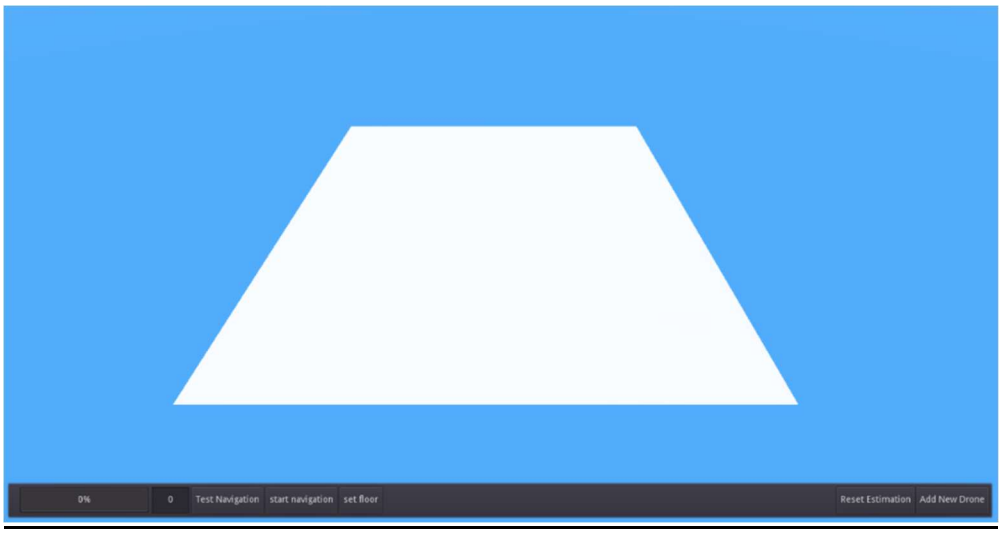


**Figura 5.3 – scena principale di Godot**

## 5.4 - Connessione dei droni

Per connettere un nuovo drone basta premere il tasto in basso a destra ***Add New Drone*** e si aprirà la finestra in figura 5.4, l'uri preimpostato è quello del drone usato per i test durante lo sviluppo del progetto.


**Figura 5.4 –finestra di connessione**

Da questa finestra è possibile connettersi a un drone scrivendo il suo uri nella casella di testo e successivamente premendo sul tasto ***connect to uri***, oppure è possibile chiudere la finestra e bloccare l'operazione di connessione premendo sul tasto ***abort*** in alto a destra.

Connesso un drone avverrà un breve caricamento, figura 5.5, che nel caso del primo drone sarà leggermente più lungo dal momento che verranno anche caricate le informazioni relative alle ancore.

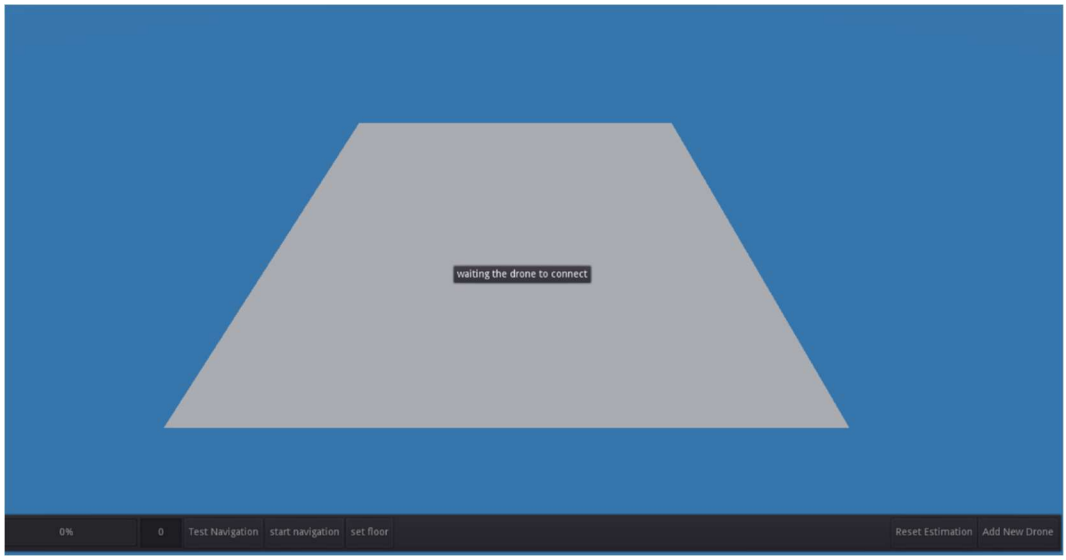


**Figura 5.5 –schermata di attesa**

Completata l'operazione di inserimento verrà mostrata una scena simile a quella in figura 5.6 dove i punti rossi agli angoli dell'ambiente sono le ancore correttamente connesse.

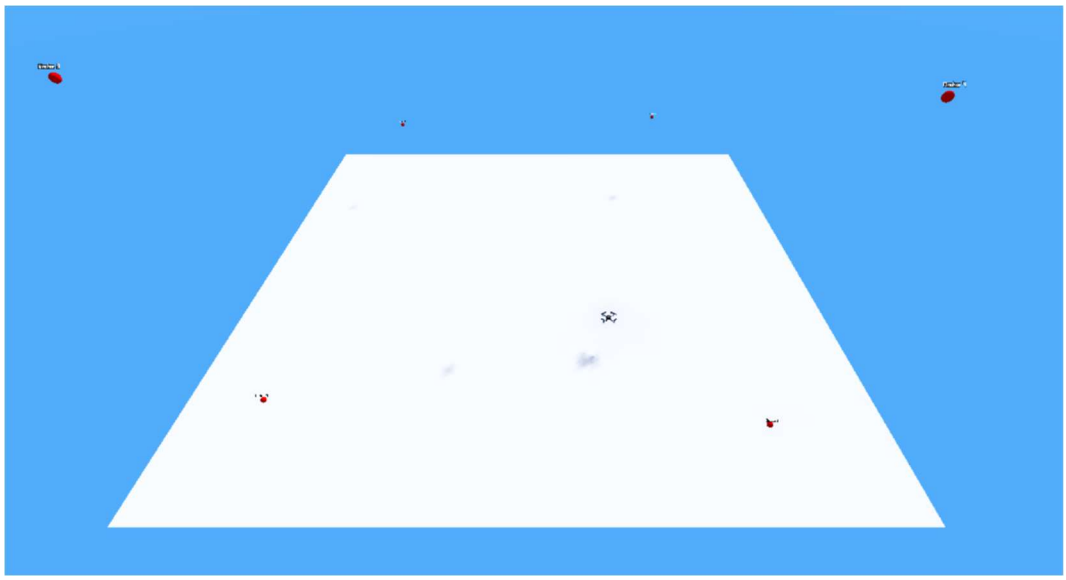

**Figura 5.6 – drone connesso**

Per ogni drone connesso si possono notare due rappresentazioni dello stesso. Come spiegato nella sezione dedicata all'oggetto touchobject, uno rappresenta quello reale ed uno quello simulato.

Nel caso presentato i 2 droni usati hanno come uri: “radio://0/90/2M/E7E7E7E702” e “radio://0/102/2M/E7E7E7E705”. Una volta connessi i droni, dopo dieci secondi, verrà effettuata una sincronizzazione tra la posizione reale e quella del drone simulato. Ci si trova dunque in una situazione simile a quella in figura 5.7.

**Figura 5.7 – entrambi i droni connessi**

## 5.5 - Creazione dei percorsi e test

Una volta connessi tutti i droni è possibile selezionarli per creare i percorsi che dovranno seguire. Si procede, dunque, aggiungendo una alla volta i punti del percorso, per farlo, dalla hud del drone selezionato si clicca sul tasto ***add point***. Come già spiegato nella sezione dedicata a Godot, la scena cambierà in una con visuale dall’alto che sarà simile a quella i figura 5.8.

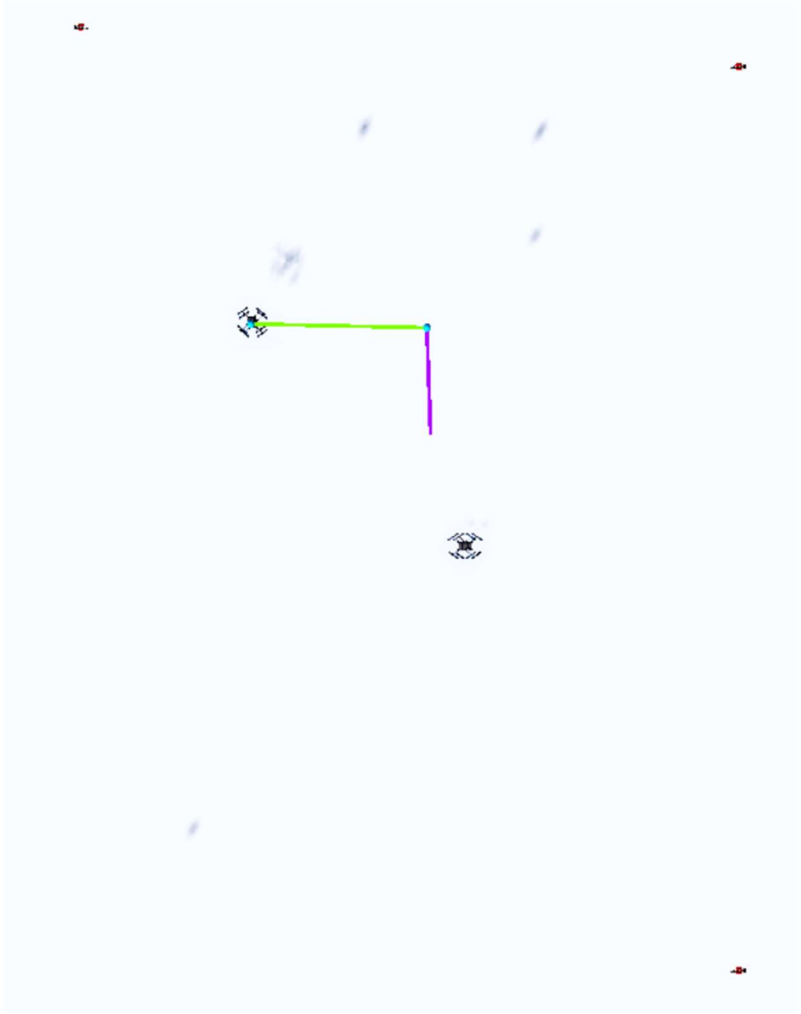

**Figura 5.8 – scena di aggiunta dei punti**

Una volta aggiunti tutti i punti che i droni dovranno eseguire, ricordandosi di inserire un punto di decollo con il tasto ***add take off point*** e il punto di atterraggio con la pressione dalla stessa gui del tasto ***add landing point***, va premuto il tasto ***Save point*** e la scena cambierà nella *SelectionState* come descritto nella disamina sul client.

Creati tutti i percorsi saremo davanti una scena simile a quella in figura 5.9.

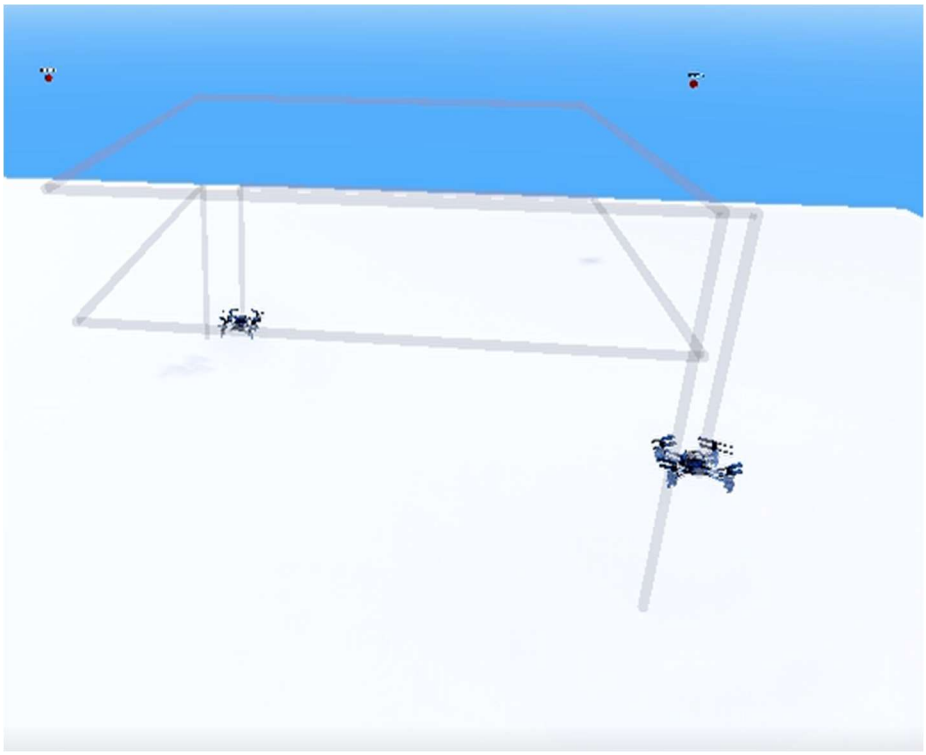

**Figura 5.9 – percorsi droni**

Una volta creati i percorsi è possibile selezionare un drone e verrà evidenziato il percorso che esso dovrà eseguire, come in figura 5.10. Da qui è possibile interagire con i punti del percorso, le sfere blu, per modificarne la posizione o eventuali comportamenti particolari come già visto nella sezione della GUI dei punti.

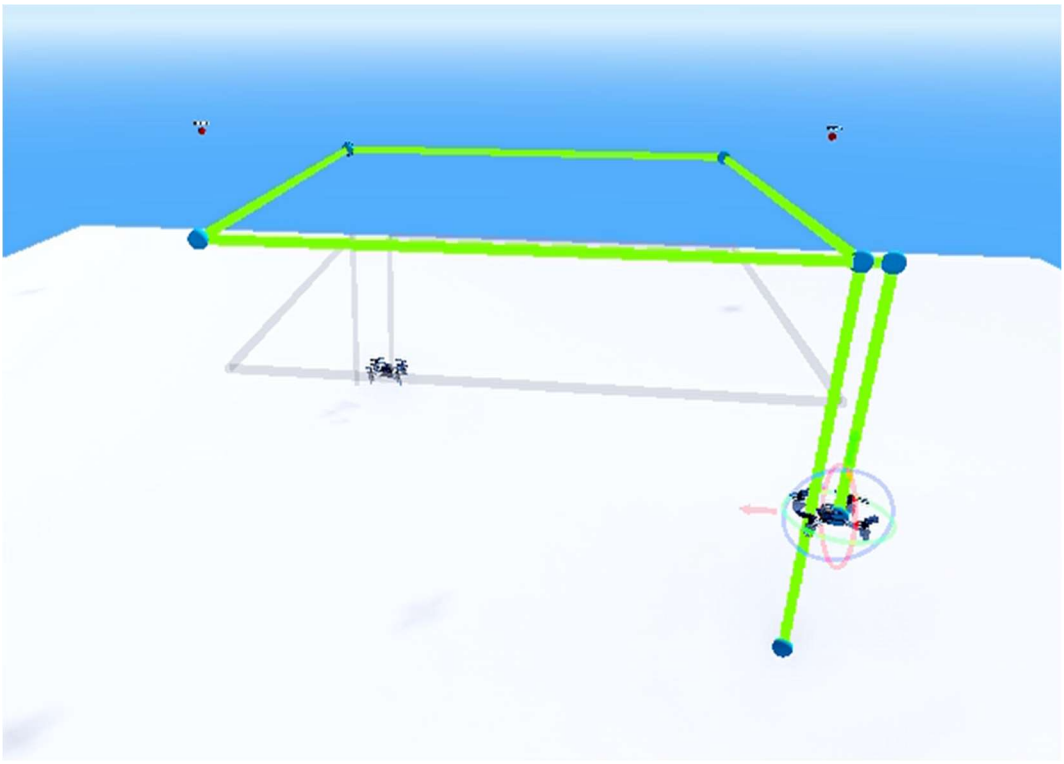

**Figura 5.10 – percorso evidenziato**

In qualsiasi momento durante la creazione del percorso è possibile testarlo cliccando sul bottone ***test navigation.*** Tramite questa funzione è possibile verificare come viene eseguita la simulazione e se causa collisioni. Nel caso in cui si causasse una collisione, la simulazione viene fermata e i droni rimangono nella posizione in cui si trovavano nell'istante in cui hanno colliso.

Come è stato detto nell'introduzione di questa sezione si vuole che un drone arrivato a un vertice del quadrato aspetti che l'altro drone arrivi al vertice opposto. A tale fine si utilizzano i meeting point. Quindi, ad ogni vertice, tranne il primo (takeoff), si assegna la proprietà di meeting point scegliendo come id di sincronizzazione "vertice 1" per il primo punto, "vertice 2 " per il secondo punto, e così via.

## 5.6 - Avvio nel reale

Dopo aver testato il percorso e averne visualizzato il corretto funzionamento, è possibile avviare il percorso nel mondo reale e monitorarlo dalla schermata di Godot, il risultato finale dell'esempio qui presentato è reperibile in [16].

# Capitolo 6 - Conclusioni

L'obbiettivo di questo progetto è quello di realizzare un software per l'orchestrare uno sciame di droni, permettendo di prevedere eventuali collisioni durante la fase di test, riuscendo a far eseguire il percorso ai droni nel mondo reale e controllando il movimento dei droni dal simulatore stesso. Ci si prefigge inoltre di raggiungere l'obiettivo utilizzando software più leggeri rispetto a Gazebo e Ros.

Tali obbiettivi sono stati pienamente soddisfatti e il risultato finale è un manager per sciami di droni in grado di effettuare la pianificazione e il controllo del volo. Inoltre, grazie all'architettura realizzata, il client e il server sono pienamente indipendenti. Ciò comporta che anche cambiando il modo in cui i droni stimano la propria posizione o utilizzando un modello diverso di droni, il software lato client continuerebbe a funzionare ugualmente. Allo stesso modo se lato client si cambiasse il modo in cui viene simulato il percorso, l'applicativo funzionerebbe lo ugualmente.

Questo tool potrebbe essere utilizzato per pianificare il volo di droni sia in ambiente interni che esterni in una moltitudine di scenari come quelli già visti nell'introduzione e da qualsiasi distanza

Tale progetto si prefigge inoltre di implementare in futuro numerose funzionalità come una gestione di ostacoli e la possibilità di generare dei percorsi per i droni a gruppi in modo che un insieme di droni faccia un movimento coordinato e sincronizzato.

# Riferimenti


[1] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, and L. Meier, "*Vision-Controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-Denied Environments,*" IEEE Robotics Automation Magazine, vol. 21, no. 3, pp. 26–40, 2014.

[2] M. A. Stuart, L. L. Marc, and J. C. Friedland, "*High Resolution Imagery Collection for PostDisaster Studies Utilizing Unmanned Aircraft Systems,*" Photogrammetric Engineering and Remote Sensing, vol. 80, no. 12, pp. 1161–1168, 2014.

[3] D. Erdos, A. Erdos, and S. E. Watkins, "*An experimental UAV system for search and rescue challenge,*" IEEE Aerospace and Electronic Systems Magazine, vol. 28, no. 5, pp. 32–37, 2013.

[4] K. Kanistras, G. Martins, M. J. Rutherford, and K. P. Valavanis, "A survey of unmanned aerial vehicles (UAVs) for traffic monitoring," in International Conference on Unmanned Aircraft Systems, 2013, pp. 221–234.

[5] A. El Saddik, *"Digital Twins: The Convergence of Multimedia Technologies,"* in IEEE MultiMedia, vol. 25, n. 2, April 2018, pp. 87–92, DOI:10.1109/MMUL.2018.023121167, ISSN 1070-986X (WC · ACNP)

[6] Mike Bacidore, *« digital twin to enable asset optimization,»* [online]. Avaiable: https://www.smartindustry.com/tools-of-transformation/data-analytics/article/11305628/digital-twin-to-enable-asset-optimization.

[7] Sito ufficiale [online]. Avaiable: https://www.ros.org/

[8] Giuseppe Silano and Luigi Iannelli, « *CrazyS: a software-in-the-loop simulation platform for the Crazyflie 2.0 nano-quadcopter*» *Robot Operating System (ROS): The Complete Reference* (Volume 4), K. Anis, Ed. Springer International Publishing, 81-115, Chapter 4, 2019.

[9] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multirobot simulator," in IEEE International Conference on Intelligent Robots and Systems, 2004, pp. 2149–2154.

[10] H. Shokry and M. Hinchey, "*Model-Based Verification of Embedded Software,*" Computer, vol. 42, no. 4, pp. 53–59, 4 2009

[11] A. Aminzadeh, M. Atashgah, and A. Roudbari, "*Software in the loop framework for the performance assessment of a navigation and control system of an unmanned aerial vehicle,*" IEEE Aerospace and Electronic Systems Magazine, vol. 33, no. 1, pp. 50–57, 1 2018.

[12] Bitcraze AB, "Crazyflie official website." [Online]. Available: https://www.bitcraze.io/

[13] Bitcraze AB, "*Loco positioning system*" [Online]. Available: https://www.bitcraze.io/documentation/system/positioning/loco-positioning-system/

[14] Jiang ZHAO, Jiaming SUN, Zhihao CAI, Yingxun WANG, Kun WU, "*Distributed coordinated control scheme of UAV swarm based on heterogeneous roles*" [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1000936121000534#f0010

[15] Bitcraze AB, "*installation*" [Online]. Available: https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/installation/install/

[16] Francesco Cristoforo Conti, "*esempio Twinflie*" [Online video]. Available: https://www.youtube.com/channel/UC_gwwztAPon2hlmsY7RAjiQ

[17] Francesco Cristoforo Conti, Federico Fausto Santoro, Corrado Santoro "Twinflie git hub repo" [Online]. Available: https://github.com/UniCT-ARSLab/Twinflie