|  |  |
| --- | --- |
| Verbale n.3 CCdS_LM4_15-03-2018_approvato web | **Corso di laurea magistrale in Ingegneria Informatica**  **A.A. 2021/2022**  **Robotica**  **Docente Prof. Antonio Chella**  **Studenti**  **Luca La Barbera**  **Salvatore Drago**  **Andrea Vaiuso** |
|  |  |

Sommario

[Introduzione 2](#_Toc95176900)

[Idea Proposta 2](#_Toc95176901)

[Generalità del progetto 2](#_Toc95176902)

[Il Robot 3](#_Toc95176903)

[Mavic 2 Pro 3](#_Toc95176904)

[Equipaggiamenti e Sensori 5](#_Toc95176905)

[Supervisor 5](#_Toc95176906)

[Progettazione del sistema 5](#_Toc95176907)

[Requisiti Funzionali 5](#_Toc95176908)

[Requisiti non Funzionali 6](#_Toc95176909)

[Modalità di Programmazione 6](#_Toc95176910)

[Cinematica 6](#_Toc95176911)

[FSDs (Finite States Diagrams) 7](#_Toc95176912)

[Legenda diagrammi a stati 7](#_Toc95176913)

[Consegna Pacchi 8](#_Toc95176914)

[Verifica Anomalia 9](#_Toc95176915)

[Implementazione del sistema 9](#_Toc95176916)

[Implementazione controllore 9](#_Toc95176917)

[utils.py 9](#_Toc95176918)

[avoid\_obstacles.py 9](#_Toc95176919)

[score\_calculator.py 10](#_Toc95176920)

[base\_controller.py 11](#_Toc95176921)

[supervisor\_controller.py 11](#_Toc95176922)

[Moto del drone 11](#_Toc95176923)

[Verifica della Stabilizzazione 14](#_Toc95176924)

[Evita ostacoli 15](#_Toc95176925)

[Comunicazione dei robot 16](#_Toc95176926)

[Notifica anomalia 17](#_Toc95176927)

[Verifica Anomalia 18](#_Toc95176928)

[Simulazione 18](#_Toc95176929)

[Ambiente di simulazione e mondo 18](#_Toc95176930)

[Problematiche di simulazione 18](#_Toc95176931)

[Conclusioni 19](#_Toc95176932)

[Limiti condizioni tecniche 19](#_Toc95176933)

[Sviluppi futuri 19](#_Toc95176934)

[Bibliografia 20](#_Toc95176935)

# Introduzione

## Idea Proposta

L’idea del progetto nasce da un radicale cambiamento nella società moderna nel modo di fare acquisti.

Oggi le persone tendono spesso ad ordinare la merce tramite le principali piattaforme di e-commerce come per esempio: Amazon, Alibaba, Zalando, etc.

L’incremento esponenziale degli ordini da consegnare e la continua ricerca da parte delle aziende di riuscire a soddisfare il cliente nel minor tempo possibile, portano le stesse a cercare nuovi strumenti che possano velocizzare e automatizzare al massimo il lavoro.

La nostra proposta è quella di unire la robotica con il mondo della delivery. Riteniamo che l’utilizzo dei droni per la consegna degli ordini possa essere di grande aiuto per le aziende, diminuendo il carico di consegne per corriere, garantendo una più elevata soddisfazione del cliente.

## Generalità del progetto

L’idea sviluppata è quella di una squadra di droni: Sky Droppers, che gestisce la consegna gli ordini.

L’intenzione del nostro progetto è quella di simulare uno scenario reale in cui gli ordini vengono disposti in delle basi di deposito, dove poi verranno presi e trasportati da un drone per essere consegnati a destinazione. Nel nostro progetto il ruolo di Order Maker è affidato al Supervisor, che dopo aver generato un ordine si occupa di comunicare in broadcast a tutti i droni la base di deposito dove si trova il pacco e l’indirizzo di consegna dello stesso.

Una volta ricevuta la notifica dal supervisor i robot si “contendono” l’ordine per decidere chi dovrà effettuare la consegna. Ciò avviene attraverso un protocollo di comunicazione fra tutti i droni dove ognuno calcola il proprio punteggio in base a vari parametri (saranno specificati nella sezione dedicata) e lo invia a tutti gli altri droni. In questo modo ogni drone ha la classifica di tutti i punteggi per ogni ordine ed è in grado di stabilire quali ordini includere nella propria lista di ordini da consegnare.

La consegna di un ordine avviene seguendo un flusso di operazioni che comprendono sia programmazione di azioni che comportamenti del tipo stimolo-risposta.

Ogni drone all’inizio si trova nella propria stazione di ricarica e, dopo avere verificato che lo stato della batteria consenta la consegna di uno o più pacchi che gli sono stati affidati, avvia i motori e si reca nella stazione di deposito associata all’ordine, dove, dopo essersi stabilizzato sopra il pacco, atterra su di esso agganciandolo attraverso un sistema di magnetico. Una volta agganciato il pacco, il drone tramite un sistema GPS e una bussola, contemporaneamente raggiunge l’altezza di navigazione e si pone in direzione della base di consegna. Non appena raggiunta l’altezza di navigazione il drone si avvia verso la base di consegna, dove una volta arrivato si stabilizzerà in posizione per poter diminuire la sua quota fino alla quota di consegna.

Giunto alla quota di consegna il drone sgancerà il pacco che dunque poi sarà ritirato dal cliente (nel nostro mondo simulato dal supervisor). Successivamente il drone controllerà lo stato della propria batteria, verificherà se sono presenti degli ordini da consegnare nella propria coda e sceglierà se sia più opportuno andare verso la base di deposito per la consegna di un nuovo ordine o se attendere nella propria stazione di raggiungere un livello di batteria sufficiente prima di ripartire.

Durante tutta l’attività di movimento del drone è attivo un sistema del tipo stimolo-risposta che garantisce che vengano evitati gli ostacoli presenti nel percorso.

Inoltre, i processi di: consegna degli ordini in coda, contesa di quelli futuri e rilevazione delle anomalie, sono in esecuzione parallela e dunque, lo svolgimento di uno dei macro task non compromette l’altro.

È presente inoltre un sistema di fault-tolerance che assicura, se uno dei droni dovesse subire un guasto e sia dunque impossibilitato ad ultimare la consegna, che un altro drone prenda in carico l’ordine, esattamente dal punto in cui il drone si è guastato.

Attualmente abbiamo simulato tutti i comportamenti descritti precedentemente tramite il simulatore Webots, modellando un mondo ideale nel quale abbiamo incluso anche degli ostacoli che sono rappresentati da dei muri che si interpongono tra il percorso del drone e la base di consegna.

Per limitazioni nella potenza di calcolo abbiamo dovuto comporre la squadra solamente da due droni, ognuno dei quali ha la propria base di ricarica. Inoltre vi sono un totale di quattro stazioni di consegna e tre basi di deposito dei pacchi.

Abbiamo realizzato il nostro progetto avvalendoci del linguaggio: Python 3

# Il Robot

## Mavic 2 Pro

Il Mavic 2 è un drone sviluppato da DJI ed è presente in due versioni:

1. Pro (quella utilizzata da noi)
2. Zoom

Ecco alcuni dati tecnici del robot:

per maggiori dettagli consultare il sito dell’azienda:

<https://www.dji.com/it/mavic-2/info>



.



## Equipaggiamenti e Sensori

Abbiamo equipaggiato il drone con diversi sensori e strumenti utili per la realizzazione dei task:

1. 8 x sensori sonar: rispettivamente uno superiore, uno anteriore, uno a sinistra, uno a destra e infine due nella parte inferiore del robot. Uno posizionato nella parte centrale del body del drone, rivolto verso il basso, che viene sfruttato per il rilevamento della box, un altro inferiore per permettere la fase di aggancio e atterraggio. Nella parte frontale e laterale del drone sono presenti cinque ulteriori sensori per individuare gli ostacoli. I sensori sono stati aggiunti all’interno del body slot aggiungendo per ognuno un nodo “DistanceSensor” ed impostando il type a “sonar”.
2. 1 x batteria, aggiunta con il nodo “battery”
3. 1 x bussola, aggiunta con il nodo “Compass”
4. 1 x IMU. È l’unità inerziale per il calcolo delle accelerazioni per imbardata beccheggio e rollio
5. 1 x magnete aggancio. Agganciato al bodyslot tramite nodo “Connector”
6. 1 x dispositivo Emettitore per far si che il robot possa mandare i messaggi utili per la collaborazione con il resto della squadra. All’interno del body slot del drone abbiamo aggiunto un nodo “emitter”
7. 1 x dispositivo Ricevitore per la ricezione dei messaggi da parte del supervisor e degli altri droni. All’interno del body slot del drone abbiamo aggiunto un nodo “receiver”

## Supervisor

Il Supervisor è un semplice bot che non ha nessuna rappresentazione fisica nel nostro mondo.

È stato dotato semplicemente di un emettitore e di un ricevitore per lo scambio dei messaggi con la squadra di droni. Inoltre il supervisor si occupa di manipolare i nodi del mondo simulato, facendo apparire e scomparire i pacchi.

# Progettazione del sistema

## Requisiti Funzionali

1. Notifica ordine
2. Contesa/assegnazione ordine
3. Ricarica batteria
4. Raggiungi base di deposito
5. Aggancia/rilascia pacco
6. Ruota verso base di consegna
7. Raggiungi quota di navigazione
8. Raggiungi base di consegna
9. Ritorna alla stazione di ricarica
10. Evita ostacoli
11. Verifica Anomalia

## Requisiti non Funzionali

1. Un ordine viene preso in carica da un drone in base ad un algoritmo che assegna ad ogni drone un punteggio in base a varie caratteristiche es: posizione, carica della batteria, etc. L’algoritmo elegge un drone vincitore, in maniera tale da garantire che ad effettuare la consegna sarà quello nelle condizioni migliori.
2. Il drone effettua una o più consegne solo dopo avere raggiunto un certo valore di percentuale di batteria, che può variare in base al numero di ordini che deve fare fino al completamento della ricarica, dove in ogni caso il drone, se sono presenti degli ordini nella propria coda, comincia ad effettuare quante più consegne possibili.
3. Divisione del codice in moduli per garantire una buona manutenibilità dello stesso.

## Modalità di Programmazione

È stata scelta una modalità di programmazione ibrida: reattiva per quanto riguarda lo svolgimento del comportamento “evita ostacoli”, deliberativa per quanto riguarda lo svolgimento di tutti gli altri compiti.

Nel nostro caso i comportamenti di tipo stimolo-risposta interferiscono continuamente con la parte deliberativa, in quanto è sempre attivo durante ogni fase del moto il controllo dei parametri dei sensori sonar per l’attivazione della funzionalità “evita ostacoli”.

## Cinematica

La cinematica adottata è di tipo inverso, sfruttando la conoscenza della propria posizione tramite GPS, bussola, e la conoscenza a priori delle posizioni delle basi, sia la base, che di consegna che di ricarica. In questo modo vengono impostate opportunamente le velocità dei motori per il raggiungimento del goal.

I movimenti tipici del drone sono i seguenti:

1. Traslazione verticale
2. Traslazione orizzontale
3. Rotazione attorno l’asse verticale (imbardata)
4. Inclinazione rispetto all’asse orizzontale (beccheggio)
5. Inclinazione rispetto all’asse laterale (rollìo)

È opportuno sottolineare come la dinamica di movimento di un drone non comporta l’esecuzione di uno solo dei precedenti tipi di moto per volta ma, al contrario, al fine di stabilizzarne la posizione e non causare situazioni che porterebbero lo stesso a danneggiarsi, è corretto, quasi obbligatorio, far sì che, anche se in maniera impercettibile, avvengano dei micro spostamenti in tutte le direzioni del moto continuamente.

## FSDs (Finite States Diagrams)

Come anticipato precedentemente, abbiamo realizzato tre macro task principali, due dei quali implementati con una macchina a stati, l’altro tramite l’utilizzo di thread che vengono eseguiti in parallelo:

1. Consegna Pacchi
2. Contesa Ordini
3. Verifica Anomalia

Di seguito riportiamo i diagrammi di esecuzione dei macro task “Consegna Pacchi” e “Verifica Anomalia”.

### Legenda diagrammi a stati

TAC: in tutti gli altri casi

O: presente un nuovo ordine da effettuare

CB: batteria carica

QR: quota raggiunta

BR: base raggiunta

S: stabilizzato

MA: magnete attivo (pacco preso)

DR: destinazione raggiunta

RO: ostacolo rilevato

AE: atterraggio eseguito

AN: anomalia rilevata

PR: pacco raggiunto

RR: stazione ricarica raggiunta

SA: spostati avanti

C9: contatore dello stato 9(serve per permanere un certo lasso di tempo nello stato)

C8: contatore dello stato 8

C19: contatore dello stato 19

-6: si proveniva da uno stato che proveniva dallo stato 6(vale per qualsiasi numero)

1: si proveniva dallo stato 1 (vale per qualsiasi numero)

!1 : non si proveniva dalla stato 1(vale per qualsiasi numero)

&: operatore booleano «and»

!: operatore booleano «not»

or : operatore booleano «or»

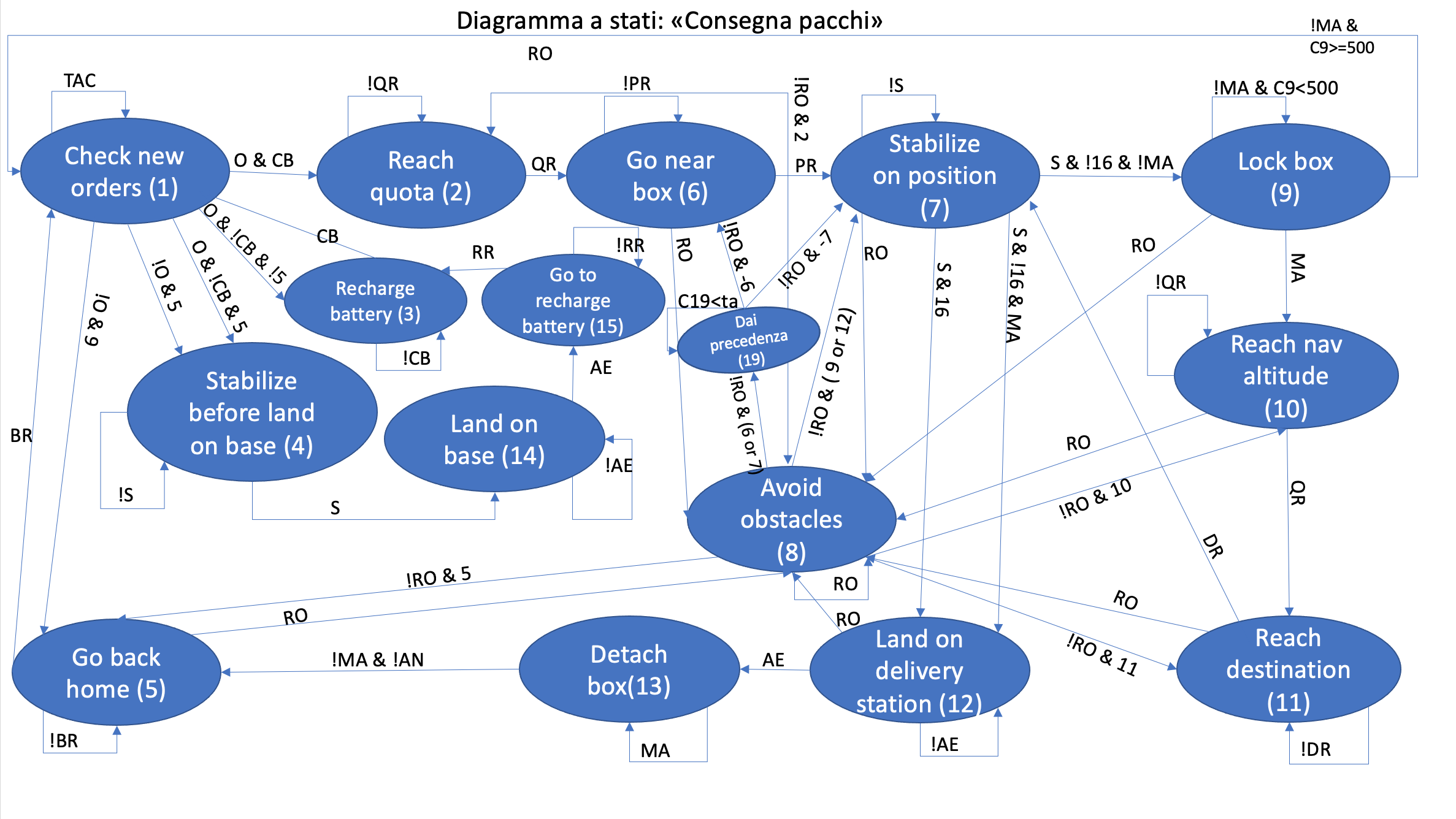
ta: tempo attesa

**Esempio:**

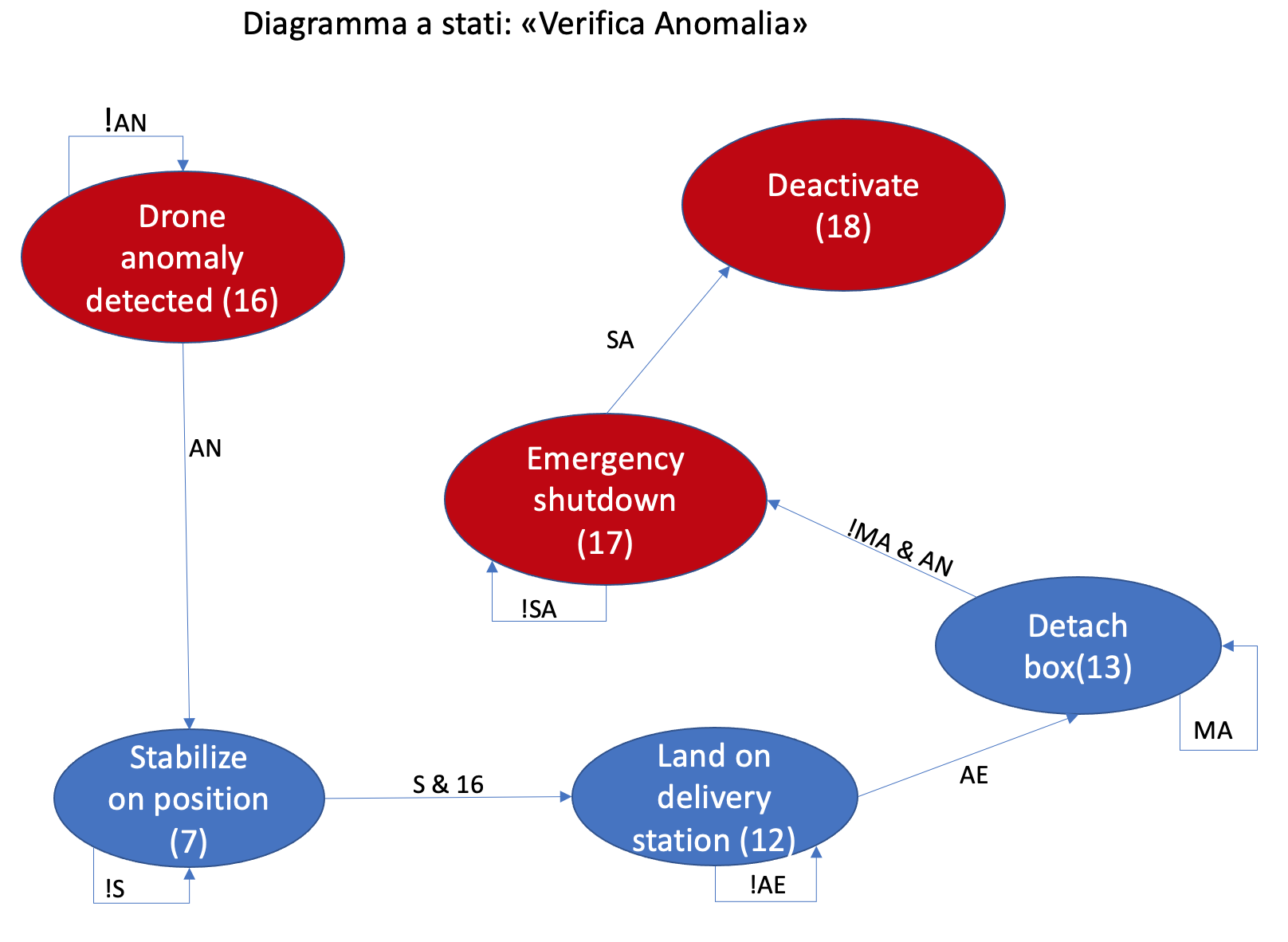
***O & !CB & !5*** 🡪 è presente un nuovo ordine da effettuare, la batteria non è carica e non si proviene dallo stato 5, ovvero da go back home, in questo caso il drone deve andare allo stato recharge battery.

***!RO & -6 🡪*** non si è rilevato nessun ostacolo e si proveniva dallo stato avoid obstacles che a sua volta proveniva dallo stato 6

### Consegna Pacchi



### Verifica Anomalia



# Implementazione del sistema

## Implementazione controllore

Abbiamo diviso il codice del controllore in come segue:

All’interno della cartella controllers vi sono due cartelle “base\_controller” e “supervisor\_controller”

* 1. In “base\_controller” troviamo i file “base\_controller.py” “utils.py” e la cartella “modules.py”
     1. All’interno di “modules” troviamo i file “avoid\_obstacles.py” e “score\_calculator.py”
  2. In “supervisor\_controller” troviamo il file “supervisor\_controller.py”

Proseguiremo il paragrafo descrivendo le funzioni implementate nei vari file:

### utils.py

In questo script si trova l’implementazione del codice di due classi:

1. StabilizationArray: classe che si occupa dell’algoritmo di stabilizzazione del drone
2. Coordinate: utile per settare e ritornare le coordinate di un drone o oggetto nel mondo

E due metodi:

1. *Euc\_dist()*: che restituisce la distanza euclidea fra due punti dati in input
2. *getID()*: che dato il nome di un drone ne restituisce l’id

### avoid\_obstacles.py

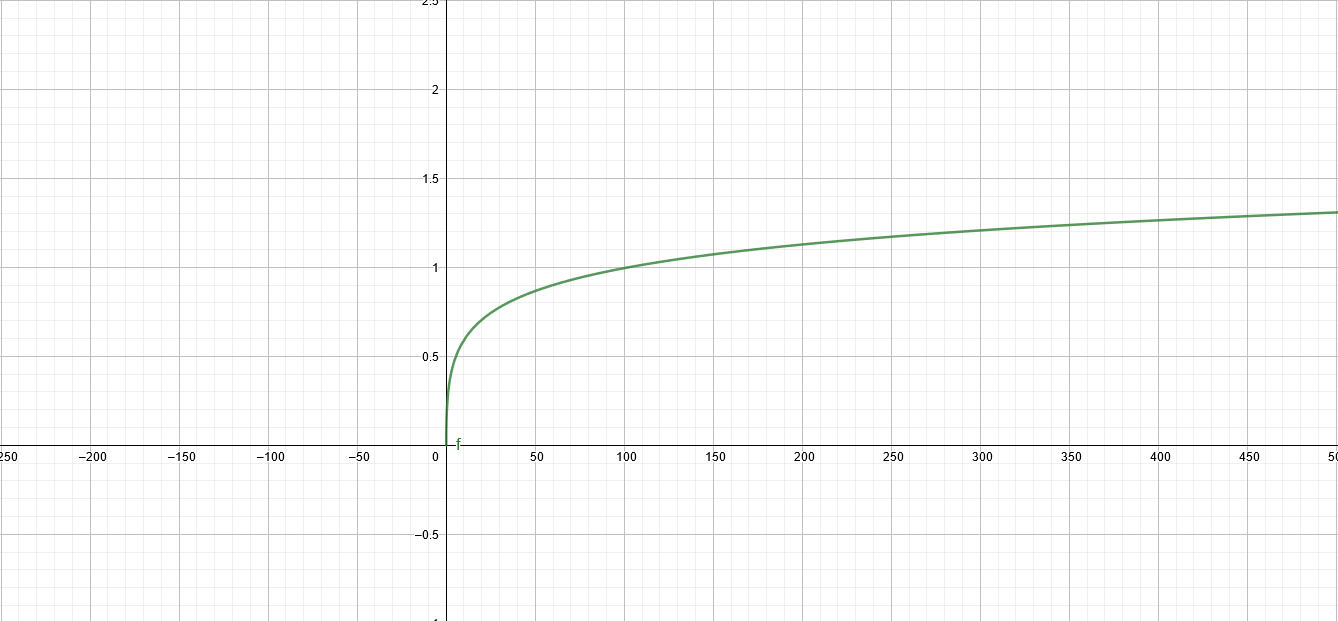
Questo script viene utilizzato per l’attivazione e la gestione del comportamento “evita ostacoli”.

Abbiamo scelto di modellare l’attivazione del comportamento con questo modulo in cui la funzione “*function(x)”* modella la sensibilità della soglia di attivazione dello stato “avoid\_obstacles” in funzione della velocità del drone. Ciò viene modellato con una funzione logaritmo, poiché riteniamo che essa approssimi in maniera ottimale e naturale il comportamento di una soglia di attivazione che è equivalente al valore massimo di distanza che può rilevare il sonar quando il drone viaggia a della velocità elevate, mentre si abbassa proporzionalmente in condizioni in cui il drone si trova a navigare a velocità medio/basse.

Con la funzione *avoid\_obstacles\_sensor(value, velocity),* attiviamo o meno la rilevazione di un ostacolo da parte di uno dei sonar.

La funzione *avoid\_obstacles\_full(front\_left, front\_right,upper\_sensor, front\_sensor,left\_sensor, right\_sensor, velocity)* ritorna “true” quando uno dei sensori rileva un ostacolo, “false” altrimenti. Viene utilizzata come funzione di controllo in tutti gli stati di movimento, per l’attivazione del comportamento “evita ostacoli”.

In basso viene mostrato il grafico della funzione logaritmo che abbiamo scelto:



### score\_calculator.py

Questo script è utilizzato per il calcolo del punteggio di un drone rispetto alla contesa di un ordine.

Questo punteggio esprime una stima in secondi del tempo necessario al drone per ultimare l’ordine corrente, tutti i suoi ordini in coda e in aggiunta quello per cui si sta calcolando il punteggio.

La funzione sccalc() prende in input la lista degli ordini in coda, l’ordine per cui si sta calcolando il punteggio (pending), l’ordine corrente, la storia degli stati, la posizione attuale del robot e le coordinate della propria stazione di ricarica. Il calcolo viene effettuato come segue:

Per tutti gli ordini da fare e quello preso in esame (pending), viene calcolata la distanza di andata e ritorno. Tale valore viene successivamente moltiplicato per una costante K=1.7, in quanto il robot percorre in media 1m in 1.7s. Sono poi aggiunte delle costanti empiriche, valutate sperimentalmente, come:

1. tempo medio di decollo
2. tempo medio di atterraggio
3. tempo medio di aggancio/rilascio box
4. tempo medio impiegato nell’evitare ostacoli lungo un tragitto

Una procedura simile viene effettuata per il calcolo del tempo che serve per concludere l’ordine corrente. Tale tempo viene calcolato in maniera differente in base alla storia degli stati e alla posizione corrente.

### base\_controller.py

Questo è il file principale per il controllore del nostro drone, dove vengono implementati quasi tutti gli stati dei diagrammi visti in precedenza. Il file è diviso in due parti principali:

1. la prima: dove vengono istanziate tutte le variabili globali di ambiente e vengono implementati gli algoritmi di calcolo dei parametri per il moto del drone, vengono anche creati i thread che gestiscono la comunicazione con la squadra di droni e il supervisor.
2. la seconda: dove viene implementato il diagramma a stati che riguarda la consegna dei pacchi e la verifica di una anomalia.

### supervisor\_controller.py

Questo è il modulo che si occupa del robot supervisore, che nel nostro mondo ha il ruolo di order maker.

## Moto del drone

Il moto del drone viene gestito con un sistema ibrido deliberativo/reattivo. La pianificazione dei vari goal segue un flusso ben preciso, e la verifica del raggiungimento degli stessi, al fine di passare da uno stato all’altro, è affidata al controllo dei valori dei dispositivi di cui è fornito il drone stesso.

A differenza di altre situazioni, la gestione del moto di un drone richiede una diversa interpretazione dei concetti di stazionarietà e posizione. Infatti, tranne nel caso in cui il robot sia fermo nella stazione di ricarica, non si avrà mai il raggiungimento preciso di una posizione esatta, ma piuttosto bisogna considerare delle posizioni che sono degli intorni di approssimazione sui quali il robot effettua una continua stabilizzazione, calcolando una certa quantità di errore nel considerare raggiunto un determinato goal.

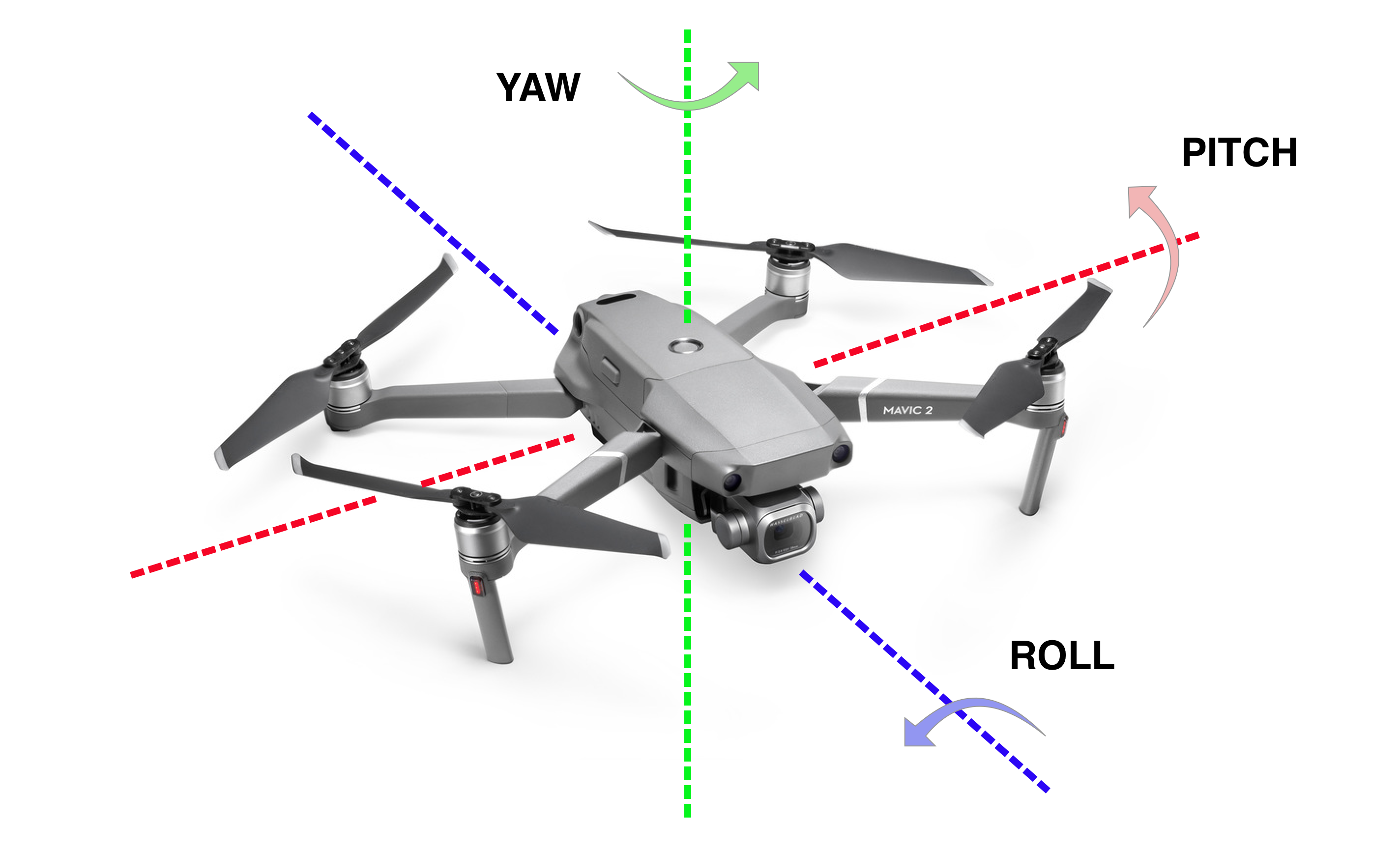
L’obiettivo che ci siamo posti è quello di far svolgere tutti i movimenti al drone in totale sicurezza, modellando uno scenario in cui i percorsi per raggiungere i goal prevedono come tipi di movimento:

1. il raggiungimento della quota prevista
2. la correzione dell’angolo rispetto al quale è rivolto il fronte del drone, allineandolo in direzione del goal
3. il raggiungimento in linea d’aria del goal
4. la stabilizzazione intorno alla posizione del goal
5. Atterraggio sul punto target

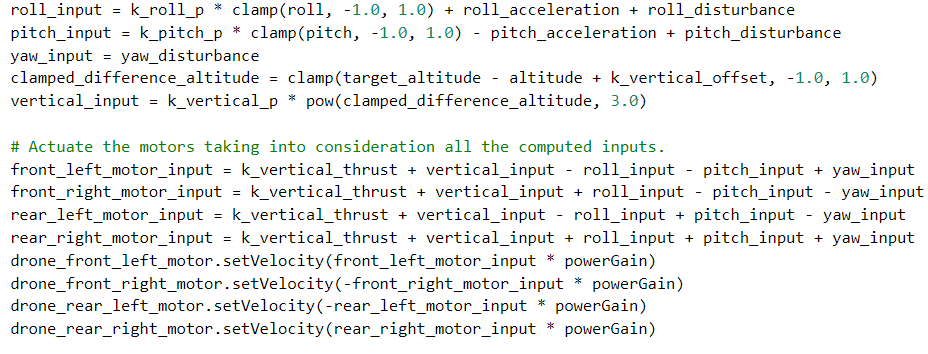
Per evitare il caso in cui un drone dovesse perdere di vista il goal e andare in un’altra direzione, il controllo per la stabilizzazione dell’angolo target, che viene fatta di continuo durante il tragitto, permetterà un reindirizzamento quasi perfetto nella direzione dello stesso.

Il drone è fornito di 4 rotori: 2 anteriori e 2 posteriori. La velocità dei quattro rotori regola il livello di imbardata, rollio e beccheggio. Il controllore agisce su un algoritmo di stabilizzazione che costringe il robot a mantenere la posizione. Il movimento è controllato dall’alterazione delle variabili:

1. Yaw\_disturbance
2. Roll\_disturbance
3. Pitch\_disturbance
4. Target\_altitude



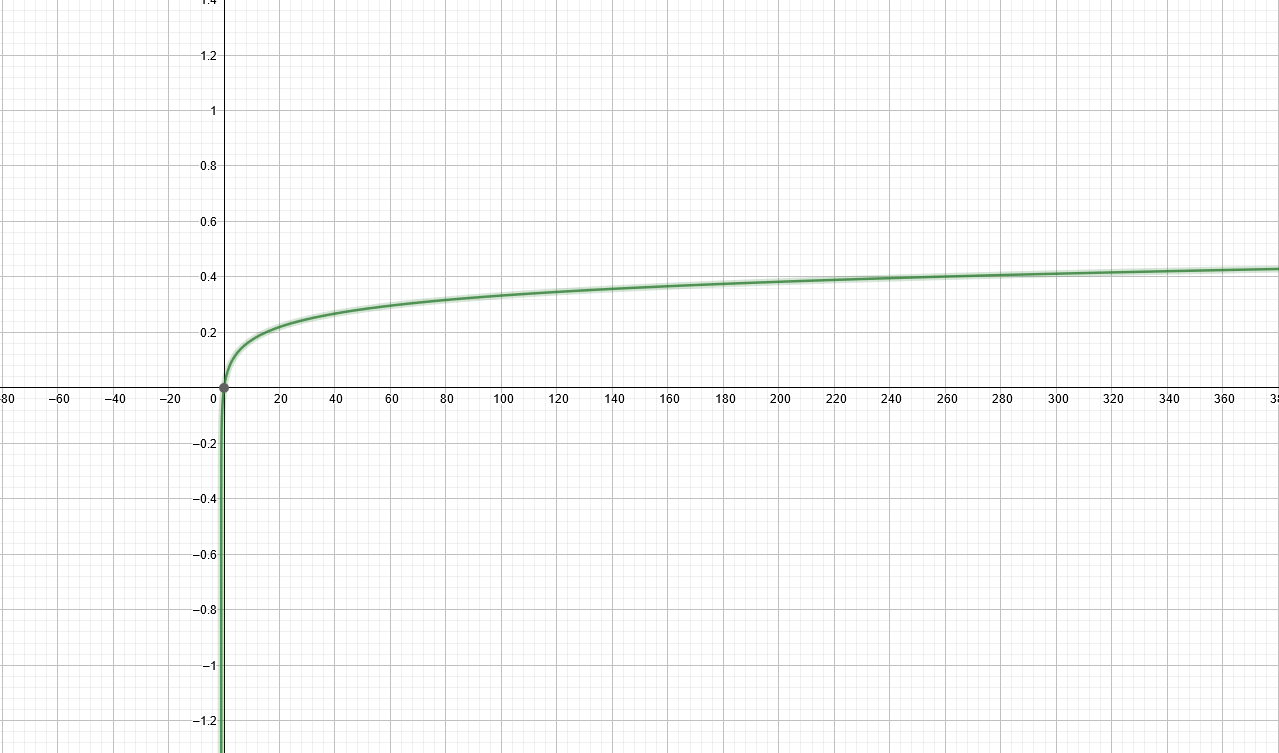
La modifica dei valori di queste variabili influenza in maniera lineare le velocità dei 4 motori che vengono calcolate come segue:



Inoltre abbiamo scelto di introdurre un valore di Gain che moltiplichiamo con i valori di velocità dei singoli rotori per agire sulla velocità di salita e discesa e per poter spegnere ed accendere il drone.

Poiché come detto precedentemente, il moto del drone comporta una continua ricerca di stabilizzazione nell’intorno di una posizione, la funzione che utilizziamo per il calcolo del beccheggio è una funzione logaritmo che ci permette di avere valori proporzionali con la distanza tra il drone e il goal desiderato, e quindi di modellare con massima precisione la velocità del drone.

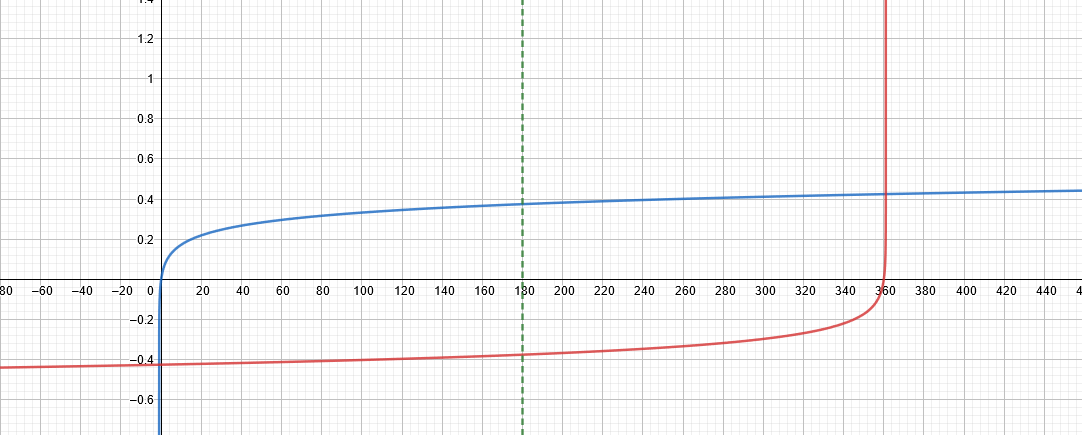
La funzione è la seguente, ne riportiamo il grafico sotto:



In particolare per quanto riguarda la yaw\_disturbance, poiché questa viene utilizzata per la rotazione attorno all’asse verticale del drone, e la distanza che viene calcolata non è fra punti nello spazio ma fra angoli, abbiamo sfruttato la f(x) di cui sopra per generare una funzione che avesse un comportamento periodico nel dominio D[0°, 360°], mantenendo sempre il principio di proporzionalità, stavolta in relazione alla differenza fra l’angolo che forma il fronte del drone con l’asse x del sistema di riferimento del mondo(bearing) e l’angolo a cui si trova il goal da raggiungere. Per calcolare tali angoli abbiamo sfruttato sia il GPS che la bussola.

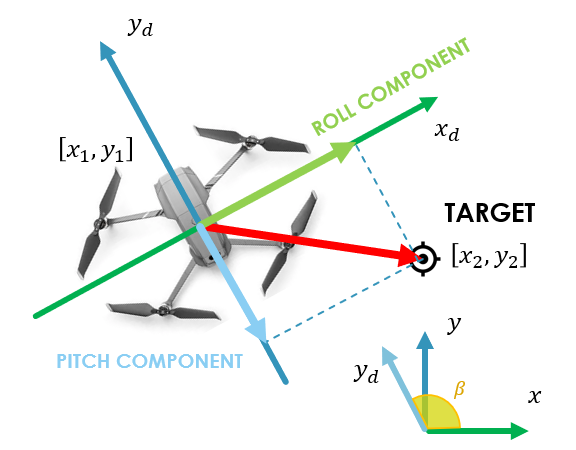
La funzione risulta dunque:

l grafico che si ottiene è il seguente:



### Verifica della Stabilizzazione

Nelle fasi di allineamento con le coordinate di destinazione, il drone deve stabilizzarsi prima di diminuire la potenza dei motori ed atterrare. La verifica del corretto stazionamento del drone è effettuata tramite il calcolo delle accelerazioni del sensore IMU in un array circolare chiamato StabilizationArray.



Il calcolo dei moduli dei valori per il rollio ed il beccheggio per ottenere una perfetta stabilizzazione è stato eseguito seguendo l’andamento di una funzione logaritmica calcolata in base alle componenti del vettore distanza calcolate rispetto all’origine traslata nel robot e ruotata nello stesso verso di esso.

### Evita ostacoli

Il comportamento evita ostacoli viene implementato interamente come un sistema di tipo stimolo-risposta. Durante il movimento del drone, in ogni stato vi è il controllo sui valori dei sonar e, qualora uno dei sensori dovesse scendere sotto il livello di soglia, la gestione del moto del drone passa allo stato “avoid\_obstacles”.

Nel nostro mondo gli ostacoli possono essere di due tipi: mobili e statici. Gli ostacoli statici nella simulazione sono rappresentati dai muri e alberi che vengo frapposti tra il percorso del robot e la base di consegna, mentre quelli dinamici per semplicità sono rappresentati dai droni stessi che durante il volo potrebbero avere delle traiettorie in comune.

In questo stato il comportamento implementato è quello di allontanarsi nella direzione opposta rispetto alla quale è stato rilevato l’ostacolo. Infatti l’implementazione consiste nella modifica delle 4 variabili: target\_altitude, yaw\_disturbance, roll\_disturbance, pitch\_disturbance.

La scelta di quale variabile modificare avviene attraverso il riconoscimento di quale sia il sensore che attiva il comportamento per evitare gli ostacoli. L’attivazione da parte di uno o più sensori fa sì che vengano modificate contemporaneamente 1 o più delle 4 variabili che gestiscono il moto del drone.

È possibile che l’ostacolo risulti rilevato, per esempio, sia dal sensore di sinistra, che da quello frontale, in questo caso vengono modificate entrambe le variabili, rispettivamente roll\_disturbance e pitch\_distrubance

Il controllo del moto ritorna alla fase deliberativa solo quando tutti i valori dei sonar risultino non rilavare alcun ostacolo.

## Comunicazione dei robot

Il protocollo di comunicazione fra i robot è il nostro secondo maggiore macro task. Sono presenti all’interno del nostro mondo due livelli di comunicazione:

1. Comunicazione tra supervisor e squadra di droni (master/slave)
2. Comunicazione tra i vari droni (peer-to-peer)

Per quanto riguarda la prima tipologia, lo scambio dei messaggi avviene al fine di notificare la presenza ai droni di un nuovo ordine che il supervisor ha generato. Inoltre vengono anche inviate le notifiche dal drone di avvenuta consegna (solo al fine di potere simulare il ritiro della box).

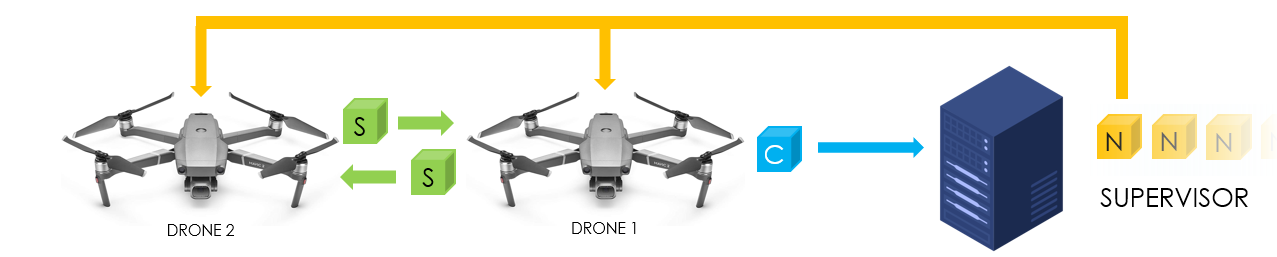
Un’altra tipologia di messaggi sono quelli inviati dai singoli droni, per comunicare al resto della squadra il proprio punteggio per il nuovo ordine appena notificato dal supervisor. Riguardo la gestione della comunicazione nei droni, abbiamo deciso di implementarla attraverso l’uso di più thread:

1. Il primo viene attivato tramite la funzione *update\_orders()*, la quale ha il compito di verificare continuamente se è presente un nuovo messaggio nella coda di arrivo del ricever del drone. Tale messaggio può essere di tipo “N”, quando siamo in presenza di una notifica di un nuovo ordine da parte del supervisor, può essere di tipo “S”, quando il messaggio ricevuto contiene il punteggio di un altro drone per un determinato ordine, oppure di tipo “C” per comunicare il completamento di un ordine al supervisor.

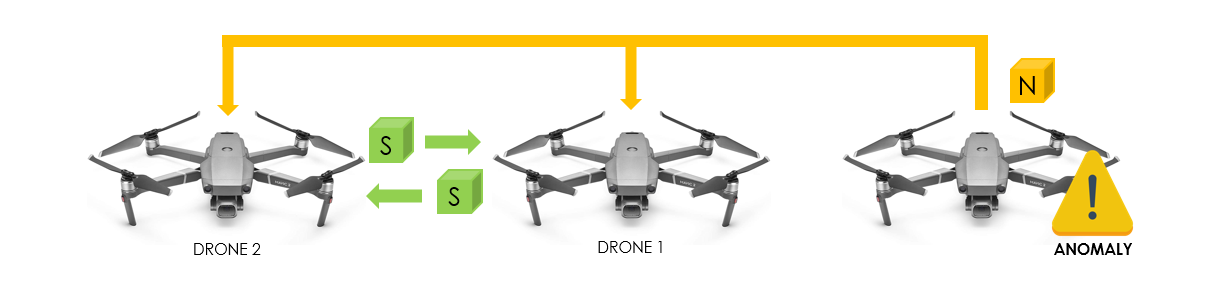
Quando siamo in presenza di un messaggio di tipo “N”, viene lanciato un’ulteriore thread tramite la funzione “send\_score()”.

Quando siamo in presenza di un messaggio di tipo “S”, viene aggiunto il punteggio che il drone ha ricevuto da un suo pari, in un dizionario che ha per chiave la coppia (Drone\_ID, Order\_ID).

1. Un altro thread viene attivato dalla funzione *send\_score(): q*uesto si occupadi richiamare la funzione *score\_calculator()* per generare il punteggio del drone, successivamente invia lo score agli altri componenti della squadra, inserisce il proprio punteggio nel dizionario e attende un tempo medio di 3 secondi dopo il quale estrae il punteggio minimo relativo all’ordine che ha causato il lancio del thread. Se il punteggio estratto è relativo al drone stesso, viene aggiunto tale nuovo ordine alla lista di quelli da eseguire. Al termine di tale compito il thread *send\_score()* termina. Sarà compito del thread *update\_orders()* generane uno nuovo per ogni notifica da parte del supervisor di un nuovo ordine.



Inoltre, in caso di anomalia, il drone può comunicare agli altri la propria posizione per il recupero di un ordine incompleto. Infine, effettuerà un atterraggio di emergenza per lasciare il pacco, in modo tale che possa essere recuperato. Nel caso in cui non vi sia un ordine in corso, il drone guasto comunicherà agli altri robot tutti gli ordini in sospeso.



Una figura esplicativa per capire la struttura dei messaggi

### 

### Notifica anomalia

Un drone in caso si verifichi una anomalia nel comportamento, dovuta a un guasto o ad un’altra qualsiasi causa, invia in broadcast un messaggio di tipo “N” agli altri droni, settando il campo “B” della base a . In questo modo le coordinate di riferimento per andare a prendere il pacco rilasciato non sono quelle di una delle basi di deposito, ma della posizione in cui il drone ha rilasciato il pacco una volta verificatasi l’anomalia, presenti negli ultimi tre campi del messaggio: APX, APY, APZ.

### Verifica Anomalia

Il verificarsi di una anomalia viene gestito nel nostro progetto seguendo dei passi ben precisi.

In prima battuta avviene il riconoscimento da parte del drone di una anomalia e immediatamente questo passa in modalità “*drone anomaly detected*”. Una volta in questo stato il drone non continua il suo percorso verso il goal ma si stabilizza nella posizione corrente. Successivamente scende fino al riconoscimento di una superficie su cui appoggiarsi. A questo punto il comportamento si diversifica in base alla presenza di un pacco trasportato o meno.

Nel caso in cui, quando si verifica una anomalia il drone stava trasportando un pacco, dopo essere atterrato, lo sgancia e invia la notifica di anomalia agli altri droni con le coordinate di dove lo ha rilasciato. In fine il drone si discosta di qualche metro per poi atterrare e spegnersi dopo aver inviato la propria lista di ordini da fare agli altri droni.

Nel caso in cui invece il drone non trasporta alcun pacco, il comportamento è quello di un semplice atterraggio di emergenza, che prevende comunque la notifica degli ordini in sospeso del drone agli altri componenti della squadra.

# Simulazione

## Ambiente di simulazione e mondo

La simulazione è stata svolta tramite il software Webots. Con l’ausilio di questo simulatore abbiamo sviluppato un mondo ideale in cui testare tutti i comportamenti sopra descritti. Abbiamo costruito un mondo dove sono state inserite 3 basi di deposito, 2 stazioni di ricarica, 2 droni, 4 basi di consegna, muri e alberi per la simulazione di ostacoli fissi.

Durante la simulazione vengono mostrati diversi scenari. I primis lo scenario classico in cui il supervisor genera un ordine, una box viene generata nella base corrispondente, avviene il protocollo di comunicazione fra i droni che decreta a chi sarà assegnata la box, una volta eletto, il drone vincitore effettuerà la consegna.

Successivamente viene mostrato anche lo scenario in cui durante la consegna di un pacco entra in gioco il comportamento “evita ostacoli”. Viene mostrato come il drone sia in grado di evitare sia ostacoli fissi (muri) sia ostacoli dinamici (altro drone).

Un altro scenario mostrato in simulazione è quello in cui il drone atterrando non trova la box nella base di deposito. Dopo aver raggiunto la quota di aggancio della box, valuterà se in caso proseguire con la consegna del prossimo ordine in coda o tornare alla stazione di ricarica.

Un altro scenario mostrato è quello della verifica dell’anomalia.

## Problematiche di simulazione

Durante le varie simulazioni che abbiamo effettuato per testare il codice ci siamo imbattuti in diverse problematiche dovute all’ambiente di simulazione.

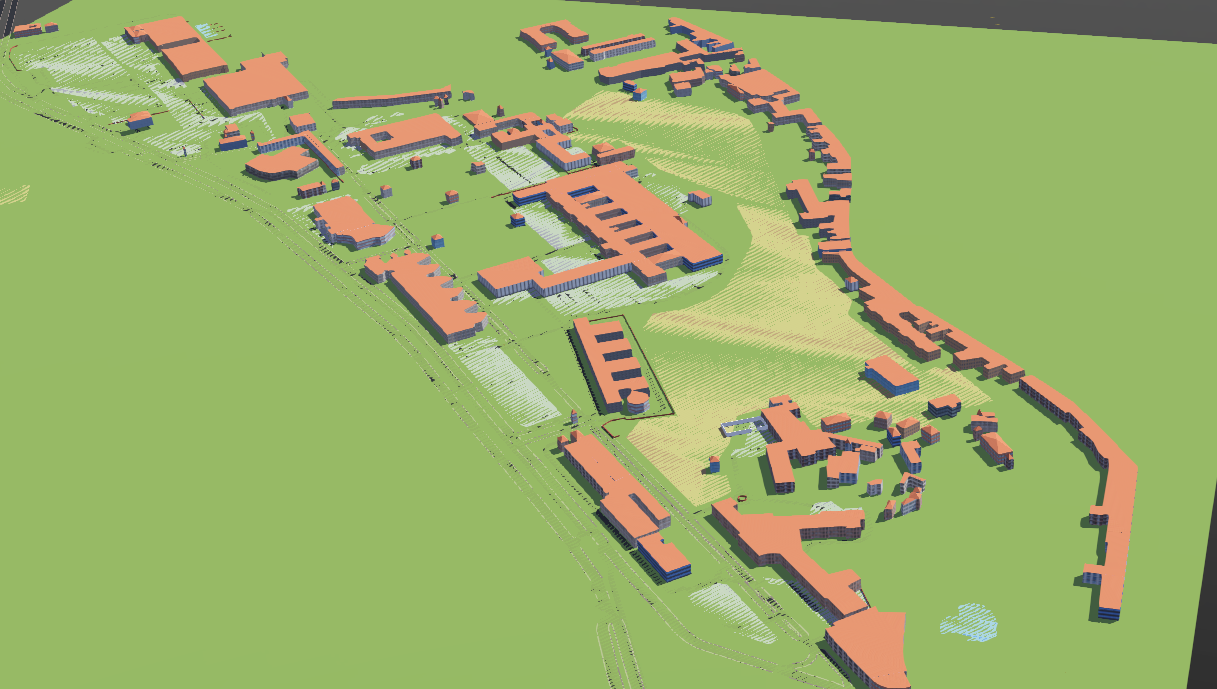
In primis, spesso capita che, in maniera assolutamente non prevedibile, vengano a crearsi nell’ambiente, indistintamente dalla posizione del drone e dal compito che sta svolgendo, dei muri invisibili che bloccano il moto del robot, o che addirittura lo fanno schiantare.

Capita a volte durante le simulazioni che dopo la generazione delle box, queste vengano spostate o che quando vengono create vengono messe in posizioni che rendono impossibile l’aggancio da parte del drone.

# Conclusioni

## Limiti condizioni tecniche

La nostra intenzione iniziale non era quella di simulare il nostro software in un mondo creato ad hoc per il nostro scopo, ma, avremmo voluto poterlo simulare nella mappa che avevamo realizzato del campus universitario di Viale delle Scienze con la presenza di più droni. Purtroppo per mancanza di sufficienti risorse di calcolo la mappa di UNIPA è stata solamente realizzata ma mai portata in simulazione poiché risultava impossibile avviare la stessa. Siamo ottimisti e magari, in futuro, con delle maggiori risorse implementeremo la simulazione nel nostro campus universitario e chissà magari un giorno proveremo il nostro software con una vera squadra di droni.



## Sviluppi futuri

Il nostro progetto si è incentrato nello sviluppo di un software per la gestione della squadra di droni. Riteniamo che questo topic riguardi non solo il mondo della robotica in quanto progressione tecnologica, ma, ha un impatto diretto su molteplici argomenti scottanti per la società moderna quali per esempio: il lavoro, l’inquinamento e il traffico.

Il lavoro svolto da una squadra di droni permette di agevolare il lavoro di ogni singolo corriere, di raggiungere in tempi più rapidi molte più destinazioni e di incentivare i clienti ad iscriversi ad abbonamenti prime per sfruttare il servizio di consegna con drone. Oggi infatti la consegna di articoli di medio/basso valore o fragilità e di contenute dimensioni, riteniamo possa essere affidata ad un drone. Amazon come azienda leader nel settore del delivery sta investendo in questa direzione con il prodotto Amazon Prime Fly https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011 .

L’impatto diretto sul lavoro svolto dai droni potrebbe incidere notevolmente sull’inquinamento e il traffico delle città. Queste potrebbero giovarne della circolazione di questi piccoli velivoli elettrici e che quindi non hanno nessun rilascio di CO2 nell’aria.

Per quanto riguarda il futuro del nostro lavoro, possibili sviluppi e modifiche:

1. si potrebbe pensare ad un sistema di objects recognition al fine di migliorare e implementare algoritmi più efficienti per quanto riguarda i comportamenti come: evita ostacoli, aggancia box, atterra nella base, ecc.
2. un altro interessante fronte di sviluppo è quello di un protocollo di comunicazione più robusto che dia certezza dell’avvenuta ricezione delle comunicazioni. Ciò permetterebbe per esempio, che alcuni comportamenti che attualmente potrebbero risultare rischiosi per il drone, vengano evitati del tutto grazie alla conferma o meno di un evento. Per esempio nel caso in cui due droni debbano consegnare nella stessa base, attualmente il conflitto viene gestito implicitamente dal comportamento evita ostacoli. Con un protocollo di comunicazione adeguato si potrebbe simulare una vera e propria torre di controllo per l’atterraggio dei droni.

# Bibliografia

<https://cyberbotics.com/doc/guide/index>

<https://stackoverflow.com/>

<https://www.dji.com/it>