Contents

1 Gitghub

Autori: Leuti Michele, Lorenzo Righi Il progetto è stato caricato sul github di ognuno degli autori al seguenti link:

- `https://github.com/MicheleLeuti/DMS/tree/master`
- `https://github.com/Raylath98/SE-Project/tree/master`

2 Descrizione Generale

Il **Drone Management System (DMS)** è progettato per gestire e monitorare efficientemente una flotta di droni per sorvegliare l'area di interesse. Il sistema è composto da diversi componenti interconnessi che lavorano insieme per garantire le prestazioni ottimali e l'affidabilità dei droni. I componenti chiave del DMS includono:

- **Centro di Controllo**: Questo è il fulcro centrale del sistema, responsabile dell'invio di istruzioni ai droni, del monitoraggio dei loro stati e della gestione della creazione e delle rotte dei droni.
- **Droni**: Questi sono le unità autonome che svolgono le missioni effettive. Ogni drone è dotato della capacità di ricevere istruzioni, seguire rotte, riportare lo stato, ricaricarsi e gestire le contingenze come batteria scarica o malfunzionamenti.
- **Monitor**: Questi sono componenti specializzati che garantiscono l'integrità, la sicurezza e l'efficienza del sistema controllando continuamente vari aspetti come i livelli di batteria, la copertura delle rotte, l'accuratezza della posizione e l'integrità dei dati.

2.1 Componenti Chiave e le loro Interazioni

- **Centro di Controllo**:
  - **Connessione al Database**: Il centro di controllo si connette a un database PostgreSQL per controllare gli stati dei droni e aggiornare informazioni sulle rotte.
  - **Gestione delle Istruzioni**: Invia istruzioni ai droni tramite un server Redis, utilizzando il canale *instruction_channel*.
  - **Monitoraggio dello Stato dei Droni**: Il centro di controllo si iscrive al canale *drone_status* su Redis per ricevere aggiornamenti in tempo reale sugli stati dei droni.
- **Droni**:
  - **Operazione Autonoma**: Ogni drone opera autonomamente, ricevendo istruzioni dal centro di controllo ed eseguendo le missioni di conseguenza.
  - **Rapporto dello Stato**: I droni riportano periodicamente il loro stato, inclusi livello di batteria, posizione e stato della missione corrente, al centro di controllo.
  - **Ricarica e Riparazione**: I droni gestiscono autonomamente la ricarica quando i livelli di batteria sono bassi e eseguono routine di auto-riparazione se necessario.
- **Monitor**:
  - **Monitoraggio della Batteria**: Questo componente controlla continuamente i livelli di batteria di tutti i droni e solleva allarmi se qualche drone ha una batteria criticamente bassa.
  - **Monitoraggio della Copertura delle Rotte**: Assicura che tutte le rotte siano coperte entro un tempo specificato. Se una rotta non viene visitata per un periodo prolungato, solleva un allarme.
  - **Monitoraggio della Posizione**: Verifica che i droni rimangano all'interno dei confini operativi designati e solleva allarmi per eventuali anomalie.
  - **Monitoraggio dell'Integrità dei Dati**: Controlla l'integrità dei dati nel database, assicurando che non ci siano duplicati, valori nulli o valori fuori intervallo.
  - **Monitoraggio dell'Integrità delle Comunicazioni**: Garantisce l'integrità dei messaggi inviati e ricevuti attraverso i canali di comunicazione Redis, verificando il formato e il contenuto delle istruzioni e degli aggiornamenti di stato.

2.2 Flusso Operativo

- **Generazione delle Rotte**:
  - Le rotte vengono generate e memorizzate nel database, ognuna con coordinate specifiche e ID unici.
- **Inizializzazione e Gestione dei Droni**:
  - Il centro di controllo inizializza i droni e assegna le rotte. Monitora continuamente gli stati dei droni e invia istruzioni secondo necessità.
- **Monitoraggio in Tempo Reale e Allarmi**:
  - I monitor funzionano in parallelo, assicurando che il sistema operi entro parametri di sicurezza ed efficienza predefiniti. Qualsiasi anomalia o problema viene immediatamente segnalato per l'azione correttiva.
- **Aggiornamenti del Database**:
  - Il centro di controllo e i droni aggiornano il database con le ultime informazioni riguardanti gli stati dei droni, le rotte coperte e eventuali problemi riscontrati.
  -

Sfruttando questi componenti e le loro interazioni, il **Drone Management System** garantisce un framework robusto e affidabile per la gestione di una flotta di droni autonomi.

3 Requisiti Utente

I requisiti degli utenti definiscono ciò che il sistema deve fare per soddisfare le esigenze degli utenti finali. Ogni requisito è numerato e descrive un singolo vincolo sul sistema. Di seguito sono elencati i requisiti utente del **Drone Management System (DMS)**:

1. **Monitoraggio in tempo reale**:

   Il sistema deve permettere il monitoraggio in tempo reale della zona di interesse.

2. **Creazione di Droni**:

   Il sistema deve permettere la creazione di nuovi droni.

3. **Assegnazione delle Rotte**:

   Il sistema deve consentire l'assegnazione di rotte specifiche ai droni.

4. **Monitoraggio in Tempo Reale**:

   Il sistema deve monitorare in tempo reale lo stato dei droni, inclusi livello di batteria, posizione e stato della missione.

5. **Gestione della Ricarica**:

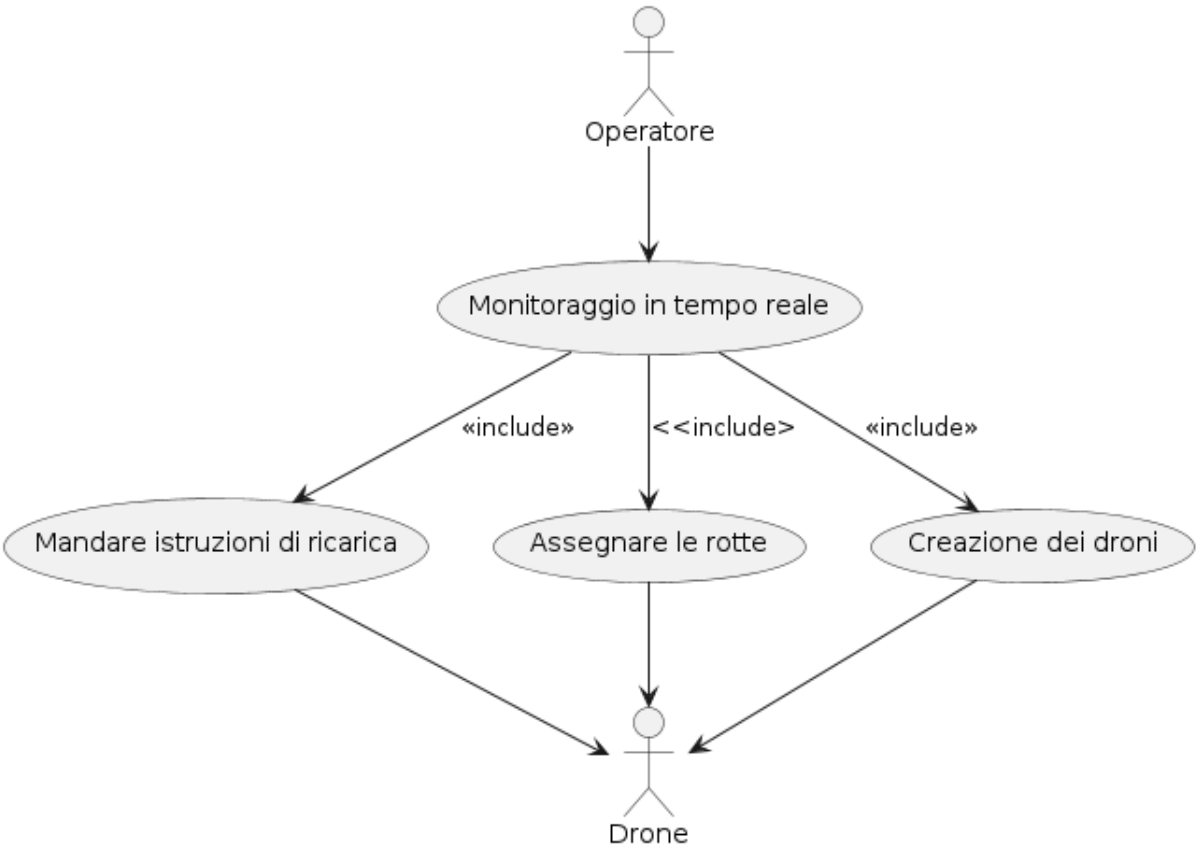
   Il sistema deve gestire la ricarica automatica dei droni quando i livelli di batteria sono bassi.

6. **Gestione database**

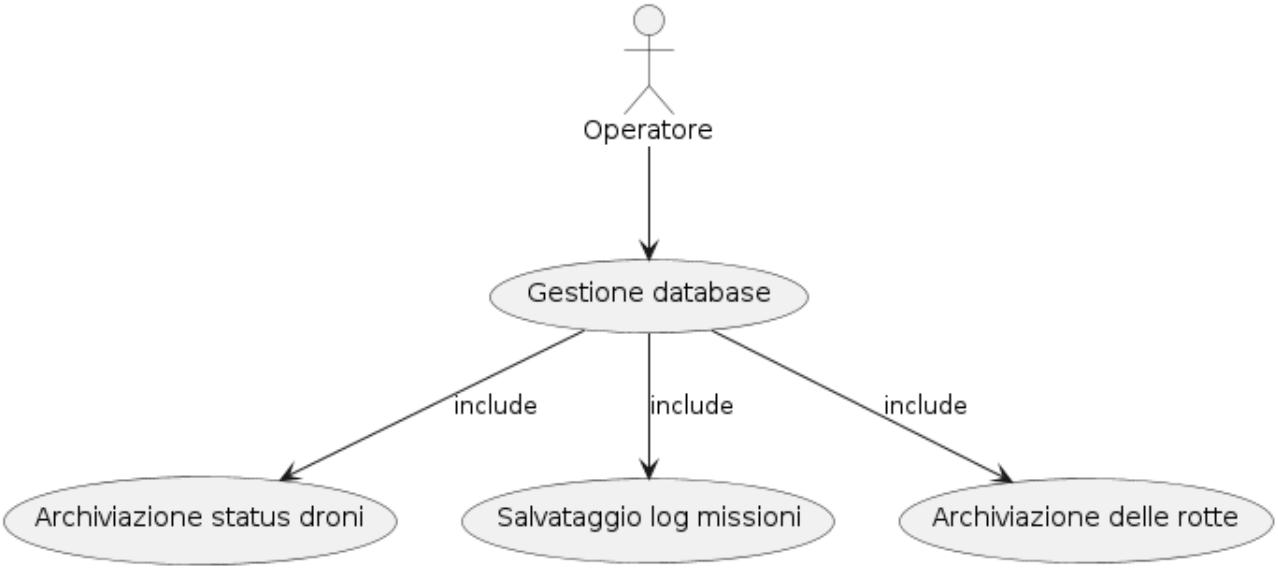
   Il sistema deve permettere di salvare le informazioni sui droni, sulle rotte e di monitorare i log delle missioni completate.

3.1 Diagramma uml

3.1.1 Monitoraggio in tempo reale



3.1.2 Gestione Database



4 Requisiti di sistema

I requisiti di sistema definiscono ciò che il sistema deve fare per soddisfare i requisiti utente. Ogni requisito è numerato e descrive un singolo vincolo sul sistema. Di seguito sono elencati i requisiti di sistema del Drone Management System (DMS):

1. **Creazione Automatica dei Droni**
   Il sistema deve creare automaticamente nuovi droni quando il numero di droni idle è inferiore al numero di rotte assegnate.

2. **Gestione delle Istruzioni**
   Il sistema deve inviare istruzioni ai droni tramite un server Redis utilizzando il canale *instruction_channel*.

3. **Monitoraggio dello Stato**
   Il sistema deve monitorare lo stato dei droni in tempo reale tramite il canale *drone_status* su Redis.

4. **Ricarica Automatica**
   Il sistema deve inviare istruzioni ai droni per ricaricarsi automaticamente quando i livelli di batteria sono bassi.

5. **Riparazione Automatica**
   Il sistema deve gestire la riparazione automatica dei droni in caso di malfunzionamenti.

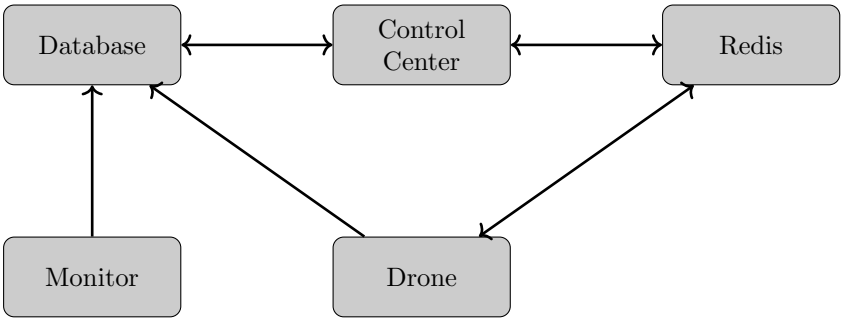
6. **Aggiornamento del Database**
   Il sistema deve aggiornare il database PostgreSQL con le informazioni più recenti riguardanti lo stato dei droni e le rotte coperte e aggiornare i log delle missioni completate.

7. **Gestione della Copertura delle Rotte**
   Il sistema deve garantire che tutte le rotte siano coperte entro un intervallo di tempo specificato.

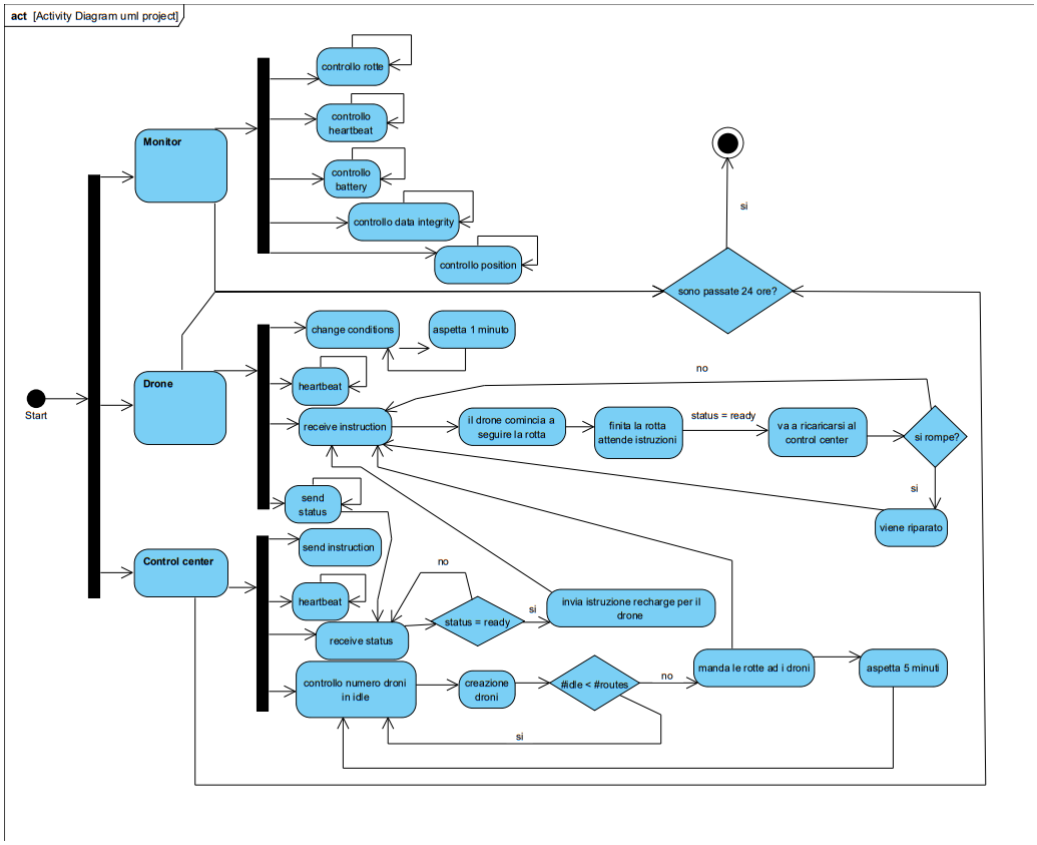
8. **Verifica della Posizione**
   Il sistema deve verificare che i droni rimangano all'interno dei confini operativi designati.

4.1 Panoramica dell'Architettura del Sistema

L'architettura del DMS può essere visualizzata come mostrato nel diagramma seguente:

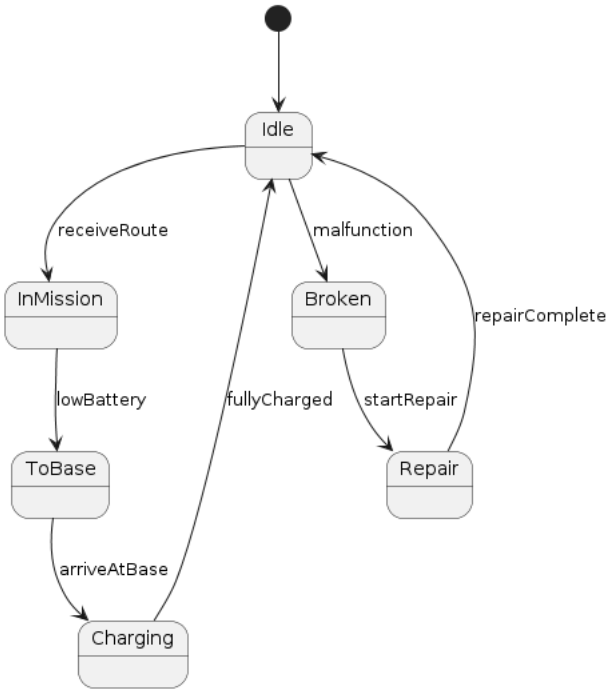


4.2 Activity Diagram UML

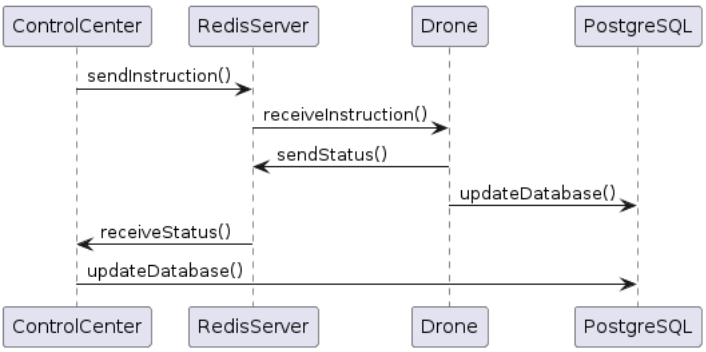


4.3 State Diagram UML

Nella seguente figura è riportato lo state diagram UML per il componente Drone:



4.4 Message Sequence Chart



5 Implementation

5.1 Pseudocodice

Pseudocodice di Control Center

```
Metodo sendInstructions
- Collegarsi al database
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- While true do
    - Contare i droni inattivi
    - If ci sono meno droni inattivi rispetto alle rotte then
        - Creare nuovi droni
    - EndIf
    - Recuperare gli ID dei droni inattivi
    - If il numero di droni inattivi è ancora insufficiente then
        - Registrare l'errore e continuare
    - EndIf
    - Inviare le rotte ai droni
    - Attendere prima di ripetere il ciclo
- EndWhile
- Chiudere la connessione al database
```

```
Metodo heartbeat
- Connettersi a Redis
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- Iscriversi al canale "heartbeat_channel"
- While true do
    - Attendere messaggi di heartbeat
    - If ricevuto un messaggio di heartbeat then
        - Rispondere con un messaggio al canale di risposta
    - EndIf
- EndWhile
- Chiudere la connessione a Redis
```

```
Metodo sendCreateInstruction
- Connettersi a Redis
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- Pubblicare il messaggio di creazione sul canale di istruzioni
- Chiudere la connessione a Redis
```

```
Metodo sendRouteInstruction
- Connettersi a Redis
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- Formattare il messaggio di istruzione con i dettagli della rotta
- Pubblicare il messaggio sul canale di istruzioni
- Chiudere la connessione a Redis
```

```
Metodo sendRechargeInstruction
- Connettersi a Redis
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- Formattare il messaggio di ricarica con i dettagli della destinazione
- Pubblicare il messaggio sul canale di istruzioni
- Chiudere la connessione a Redis
```

```
Metodo sendDroneOnRoute
- Chiamare sendRouteInstruction con i dettagli della rotta e l'ID del drone

Metodo receiveStatus
- Connettersi a Redis
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- Iscriversi al canale "drone_status"
- While true do
    - Attendere messaggi di stato dei droni
    - If ricevuto un messaggio di stato then
        - Analizzare e aggiornare lo stato del drone
        - If il drone è pronto then
            - Inviare istruzione di ricarica
        - EndIf
    - EndIf
- EndWhile
- Chiudere la connessione a Redis
```

```
Metodo updateDatabase
- While true do
    - Collegarsi al database
    - If la connessione fallisce then
        - Registrare l'errore e uscire
    - EndIf
    - Aggiornare lo stato delle rotte nel database
    - Inserire nuovi log dei droni nel database
    - Chiudere la connessione al database
    - Attendere un minuto
- EndWhile
```

```
Metodo calculateDistanceToBase
- Calcolare e restituire la distanza dalla base utilizzando le coordinate
```

```
Metodo calculateTimeToBase
- Calcolare e restituire il tempo necessario per raggiungere la base in base alla distanza
```

```
Metodo updateDroneStatus
- Aggiornare lo stato del drone nel sistema con i nuovi dati
```

```
Metodo extractDroneNumber
- Estrarre e restituire il numero del drone dall'ID del drone
```

```
Metodo printMap
- Ordinare e stampare lo stato di tutti i droni
```

Pseudocodice di Drone

```
Metodo startThreads
- If droneId è 0 then
    - Avviare il thread per sendStatus
    - Avviare il thread per printStatus
    - Avviare il thread per receiveInstruction
    - Avviare il thread per updateDatabase
    - Avviare il thread per changeConditions
    - Avviare il thread per heartbeat
- Else
    - Avviare il thread per receiveInstruction
- EndIf
```

```
Metodo changeConditions
- While true do
    - Generare un valore casuale per il vento
    - Generare un valore casuale per il tempo
    - Assegnare il valore del tempo (sunny, rainy, foggy)
    - Stampare i parametri cambiati
    - Attendere un minuto
- EndWhile
```

```
Metodo printStatus
- While true do
    - Stampare lo stato corrente dei droni
    - Attendere 10 secondi
- EndWhile
```

```
Metodo heartbeat
- Connettersi a Redis
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- Iscriversi al canale "heartbeat_channel"
- While true do
    - Attendere messaggi di heartbeat
    - If ricevuto un messaggio di heartbeat then
        - Rispondere con un messaggio al canale di risposta
    - EndIf
- EndWhile
- Chiudere la connessione a Redis
```

```
Metodo updateDatabase
- Attendere 5 secondi
- Collegarsi al database
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- While true do
    - Creare la query di aggiornamento per lo stato dei droni
    - Eseguire la query
    - If la query fallisce then
        - Registrare l'errore
    - EndIf
- EndWhile
- Chiudere la connessione al database
```

```
Metodo createNewDrone
- Generare un nuovo ID per il drone
- Generare casualmente la vita del drone
- Creare un nuovo drone e aggiungerlo alla mappa dei droni
- Incrementare il contatore dei droni inattivi
- Collegarsi al database
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- Inserire il nuovo drone nel database
- Avviare i thread del nuovo drone
- Stampare il messaggio di creazione del drone
```

```
Metodo repair
- Attendere 10 secondi
- Ripristinare la batteria del drone
- Generare casualmente il tempo di riparazione
- Stampare il messaggio di inizio riparazione
- Attendere il tempo di riparazione generato
- Generare casualmente la nuova vita del drone
- Aggiornare lo stato del drone a "idle"
- Stampare il messaggio di fine riparazione
```

```
Metodo sendStatus
- While true do
    - Per ogni drone:
        - Connettersi a Redis
        - Generare il messaggio di stato del drone
        - Pubblicare il messaggio sul canale "drone_status"
        - Chiudere la connessione a Redis
```

```
Metodo receiveInstruction
- Connettersi a Redis
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- Iscriversi al canale "instruction_channel"
- While true do
    - Attendere messaggi di istruzione
    - If ricevuto un messaggio di istruzione then
        - If istruzione è "create_drone" then
            - Creare un nuovo drone
        - Else
            - Estrarre i dettagli dell'istruzione
            - Chiamare followInstruction con i dettagli dell'istruzione
        - EndIf
    - EndIf
```

```
Metodo followInstruction
- If l'ID dell'istruzione non corrisponde al drone then
    - Uscire
- EndIf
- If il tipo è "recharge" then
    - Estrarre le coordinate di destinazione
    - If lo stato del drone è "ready" then
        - Aggiornare lo stato del drone a "to_base"
        - Chiamare moveToDestination con le coordinate
    - EndIf
- Else if il tipo è "follow_route" then
    - If lo stato del drone non è "idle" then
        - Uscire
    - EndIf
    - Estrarre i punti della rotta
    - Aggiornare lo stato del drone a "in_mission"
    - Chiamare followRoute con la rotta
- EndIf
```

```
Metodo moveToDestination
- While la distanza alla destinazione è maggiore della visibilità do
    - Calcolare la distanza alla destinazione
    - Calcolare il passo di movimento
    - Aggiornare le coordinate del drone
    - Ridurre la batteria del drone
    - If la batteria è scarica then
        - Aggiornare lo stato del drone a "broken"
        - Chiamare repair
    - EndIf
    - Attendere 1 millisecondo
- EndWhile
- If lo stato è "to_base" then
    - Ridurre la vita del drone
    - If la vita è zero then
        - Aggiornare lo stato a "broken"
        - Chiamare repair
    - Else
        - Aggiornare lo stato a "recharge"
        - Chiamare recharge
    - EndIf
- EndIf
```

```
Metodo followRoute
- If lo stato è "in_mission" then
    - Per ogni punto della rotta:
        - Chiamare moveToDestination con il punto
    - Aggiornare lo stato del drone a "ready"
- EndIf
```

```
Metodo recharge
- Generare casualmente il tempo di ricarica
- Stampare il messaggio di ricarica
- While il tempo di ricarica è maggiore di zero do
    - Attendere 1 millisecondo
    - Ridurre il tempo di ricarica
- EndWhile
- Ripristinare la batteria del drone
- Aggiornare lo stato del drone a "idle"
```

```
Metodo distanceTo
- Calcolare e restituire la distanza tra il drone e le coordinate specificate
```

5.2 Pseudocodice monitor

5.2.1 Monitor battery check

```
Metodo run
- Collegarsi al database
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- While true do
    - Connessione al database

    - Query per contare i droni con battery_seconds = 0 e status 'in_mission' o 'ready' o 'broken'
    - If la query fallisce then
        - Registrare l'errore e uscire
    - EndIf

    - Estrazione del risultato della query
    - If il conteggio è maggiore di 0 then
        - Stampare un messaggio di allarme con il numero di droni con batteria scarica
        - Attendere 10 secondi
    - EndIf
- EndWhile
- Chiudere la connessione al database
```

5.2.2 Monitor data integrity

```
Metodo run
- Collegarsi al database
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- While true do
    - Verifica di duplicati
        - Eseguire la query per trovare duplicati
        - If la query fallisce then
            - Registrare l'errore
        - Else
            - If ci sono duplicati then
                - Stampare un messaggio di errore
                - Per ogni duplicato
                    - Stampare l'ID del drone e il numero di duplicati
                    - Attendere 10 secondi
                - EndFor
            - EndIf
        - EndIf
        - Liberare la memoria della query

    - Verifica di nulli in colonne obbligatorie
        - Eseguire la query per trovare valori nulli
        - If la query fallisce then
            - Registrare l'errore
        - Else
            - If ci sono valori nulli then
                - Stampare un messaggio di errore
                - Attendere 10 secondi
            - EndIf
        - EndIf
        - Liberare la memoria della query
```

```
    - Verifica di valori fuori range
        - Eseguire la query per trovare valori fuori range
        - If la query fallisce then
            - Registrare l'errore
        - Else
            - If ci sono valori fuori range then
                - Stampare un messaggio di errore
                - Attendere 10 secondi
            - EndIf
        - EndIf
        - Liberare la memoria della query
- EndWhile
- Chiudere la connessione al database
```

5.2.3 Monitor position

```
Metodo run
- Collegarsi al database
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf
- While true do
    - Definire la query per verificare le posizioni fuori dai limiti
    - Eseguire la query
    - If la query fallisce then
        - Registrare l'errore e uscire
    - EndIf

    - Estrarre il numero di righe nel risultato
    - If ci sono righe nel risultato then
        - Per ogni riga nel risultato:
            - Estrarre l'ID del drone, posX, e posY
            - Stampare un messaggio di anomalia di posizione
            - Attendere 10 secondi
        - EndFor
    - EndIf
    - Liberare la memoria della query
- EndWhile
- Chiudere la connessione al database
```

5.2.4 Monitor route coverage

```
Metodo populateRoutes
- Collegarsi al database
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf

- Definire la query per contare le rotte
- Eseguire la query
- If la query fallisce then
    - Registrare l'errore e uscire
- EndIf

- Estrarre il numero di rotte
- If il numero di rotte è maggiore di 0 then
    - Uscire (le rotte sono già popolate)
- EndIf

- Definire la query per inserire una rotta
- For ogni valore di y da 0.01 a 6.0 con passo 0.02 do
    - Formattare il valore di y
    - Definire i parametri della query
    - Eseguire la query di inserimento
    - If la query fallisce then
        - Registrare l'errore
    - EndIf
- EndFor

- Chiudere la connessione al database
- Stampare il messaggio di successo
```

```
Metodo run
- Collegarsi al database
- If la connessione fallisce then
    - Registrare l'errore e uscire
- EndIf

- While true do
    - Definire la query per verificare le rotte non visitate negli ultimi 10 minuti
    - Eseguire la query
    - If la query fallisce then
        - Registrare l'errore e continuare
    - EndIf

    - Estrarre il numero di righe nel risultato
    - For ogni riga nel risultato do
        - Estrarre posX e posY
        - Stampare un messaggio di allarme per la rotta non visitata
        - Attendere 10 secondi
    - EndFor

    - Liberare la memoria della query
- EndWhile

- Chiudere la connessione al database
```

5.2.5 System availability monitor

```
Costruttore SystemAvailabilityMonitor
- Connettersi a Redis
- If la connessione fallisce then
    - Registrare l'errore
- EndIf
```

```
Distruttore ~SystemAvailabilityMonitor
- If context è non nullo then
    - Liberare context
- EndIf
```

```
Metodo run
- Attendere 20 secondi
- Avviare il thread per sendHeartbeat
- Avviare il thread per receiveHeartbeat
- Avviare il thread per checkComponentStatus
- Attendere la terminazione dei thread
```

```
Metodo sendHeartbeat
- While true do
    - Inviare il messaggio di heartbeat a Redis
    - If invio fallisce then
        - Registrare l'errore
    - EndIf
    - Attendere 5 secondi
- EndWhile
```

```
Metodo receiveHeartbeat
- Connettersi a Redis come subscriber
- Iscriversi al canale "heartbeat_response_channel"
- If la connessione fallisce then
    - Registrare l'errore
- EndIf

- While true do
    - Attendere i messaggi di heartbeat di risposta
    - If il messaggio è valido then
        - Aggiornare l'ultimo heartbeat per il componente
    - EndIf
- EndWhile

- Liberare la connessione subscriber
```

```
Metodo checkComponentStatus
- While true do
    - Ottenere il tempo corrente
    - For ogni componente in lastHeartbeat do
        - Calcolare la durata dall'ultimo heartbeat
        - If la durata è maggiore di 10 secondi then
            - Stampare un messaggio di errore
        - Else
            - Stampare un messaggio di disponibilità
        - EndIf
    - EndFor
    - Attendere 60 secondi
- EndWhile
```

5.3 Schema database

```
CREATE TABLE IF NOT EXISTS drone (
    id SERIAL PRIMARY KEY,
    drone_id VARCHAR(50),
    status VARCHAR(50),
    battery_seconds INT,
    pos_x DOUBLE PRECISION,
    pos_y DOUBLE PRECISION,
    log_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS routes (
    id SERIAL PRIMARY KEY,
    pos_x DOUBLE PRECISION,
    pos_y DOUBLE PRECISION,
    visited BOOLEAN DEFAULT FALSE,
    visited_time TIMESTAMP
);

CREATE TABLE IF NOT EXISTS drone_logs (
    id SERIAL PRIMARY KEY,
    descrizione VARCHAR(255),
    log_time TIMESTAMP
);
```

- **Tabella** `drone`
  - `id`: Chiave primaria, auto-incrementante.
  - `drone_id`: Identificatore del drone, tipo stringa.
  - `status`: Stato del drone, tipo stringa.
  - `battery_seconds`: Durata della batteria in secondi, tipo intero.
  - `pos_x`: Posizione X del drone, tipo double.
  - `pos_y`: Posizione Y del drone, tipo double.
  - `log_time`: Timestamp dell'ultima registrazione, con valore predefinito l'ora corrente.
- **Tabella** `routes`
  - `id`: Chiave primaria, auto-incrementante.
  - `pos_x`: Posizione X della rotta, tipo double.
  - `pos_y`: Posizione Y della rotta, tipo double.
  - `visited`: Booleano che indica se la rotta è stata visitata, con valore predefinito `false`.
  - `visited_time`: Timestamp dell'ultima visita.
- **Tabella** `drone_logs`
  - `id`: Chiave primaria, auto-incrementante.
  - `descrizione`: Descrizione del log, tipo stringa.
  - `log_time`: Timestamp del log.

5.4 Descrizione delle connessioni Redis

Nel progetto, Redis viene utilizzato come meccanismo di messaggistica per facilitare la comunicazione tra il centro di controllo e i droni. Di seguito sono descritte le varie connessioni Redis e i canali utilizzati:

5.4.1 Connessione per inviare istruzioni

Il centro di controllo utilizza Redis per inviare istruzioni ai droni tramite il canale `instruction_channel`. Le istruzioni possono essere di tre tipi principali: `create_drone`, `follow_route` e `recharge`. Ogni tipo di istruzione ha un formato specifico.

- **create_drone**: Questa istruzione viene utilizzata per creare un nuovo drone. Il messaggio pubblicato su `instruction_channel` è semplicemente `create_drone`.
- **follow_route**: Questa istruzione contiene l'ID del drone, il tipo di istruzione, l'ID del percorso e le coordinate del percorso. Esempio di formato del messaggio:
  `drone_1:follow_route:routeId:123:0.0,2.31;6.0,2.31`.
- **recharge**: Questa istruzione contiene l'ID del drone, il tipo di istruzione, l'ID del percorso e le coordinate della destinazione di ricarica. Esempio di formato del messaggio:
  `drone_1:recharge:routeId:0`.

5.4.2 Connessione per ricevere lo stato dei droni

I droni utilizzano il canale `drone_status` per inviare periodicamente il loro stato al centro di controllo. Ogni messaggio di stato include l'ID del drone, il tempo di batteria rimanente, la posizione corrente (coordinate x e y), lo stato attuale del drone e l'ID del percorso (di default se non vi è alcuna rotta assegnata). Esempio di formato del messaggio: `drone_1:1200:4.34,5.78:idle:0`. Il

5.4.3 Connessione per l'heartbeat

Sia il centro di controllo che i droni utilizzano il canale `heartbeat_channel` per monitorare la disponibilità del sistema. Il monitor invia un messaggio `heartbeat` a cui i droni e il centro di controllo rispondono con i rispettivi messaggi di risposta (`drone` per i droni e `Control_center` per il centro di controllo) sul canale `heartbeat_response_channel`.

5.5 Risultati della Simulazione del Sistema

La simulazione del sistema di controllo dei droni ha prodotto diversi risultati significativi che sono stati analizzati per valutare l'efficacia e la robustezza del sistema. Di seguito vengono riportati i principali risultati ottenuti:

5.5.1 Creazione e Gestione dei Droni

Durante la simulazione, il sistema è stato in grado di creare nuovi droni dinamicamente in risposta alla domanda. In particolare:

- **Numero di droni creati**: Il sistema per un ora di esecuzione ha creato un totale di circa 3000 droni
- **Tempo medio di creazione**: Il tempo medio per la creazione di 300 droni è di circa 15 secondi.

5.5.2 Esecuzione delle Missioni

Le missioni sono state assegnate e completate dai droni in modo efficiente. I risultati specifici includono:

- **Numero di missioni completate**: per un ora di esecuzioni circa 2700 missioni completato con successo
- **Tempo medio per missione**: Il tempo medio per completare una missione è stato di 15 minuti.
- **Percentuale di successo**: Il tasso di successo delle missioni è stato del 100%.

5.5.3 Monitoraggio e Risposte del Sistema

Il sistema di monitoraggio ha rilevato e risposto efficacemente a vari eventi. In particolare, sono stati effettuati test appositi per far attivare i monitor e ognuno ha funzionato egregiamente, in particolare:

- Abbiamo assegnato ai droni una posizione non valida (non nell'area di sorveglianza). Il monitor ha rilevato subito l'anomalia.
- Abbiamo fatto in modo di inserire dati non validi nel database. Il monitor ha rilevato subito l'anomalia.
- Durante l'esecuzione abbiamo stoppato il processo dei droni, mentre il control center era ancora in esecuzione. Il monitor ha rilevato subito l'anomalia.
- Abbiamo assegnato una durata di batteria molto inferiore a quella prevista per fare in modo che i droni si scaricassero durante la missione. Il monitor ha rilevato subito l'anomalia.
- Non abbiamo assegnato tutte le rotte ai droni. Il monitor ha rilevato subito l'anomalia.

5.5.4 Conclusioni

I risultati della simulazione dimostrano che il sistema di controllo dei droni è robusto e capace di gestire un elevato numero di droni e missioni con la massima affidabilità. I monitoraggi implementati hanno garantito la correttezza delle operazioni e la disponibilità del sistema, contribuendo a un elevato tasso di successo delle missioni.

Una delle principali sfide affrontate in questo progetto è stata l'integrazione e l'interazione tra tutte le componenti del sistema. La gestione di un gran numero di droni ha presentato difficoltà significative, soprattutto in termini di coordinamento e comunicazione. Tuttavia, siamo riusciti a superare questi ostacoli e a garantire una comunicazione efficace tra tutte le parti coinvolte.

Per migliorare ulteriormente l'efficienza del sistema e ridurre il numero di droni necessari, suggeriamo le seguenti strategie:

- **Migliorare l'autonomia dei droni**: Aumentare la durata della batteria per ridurre la frequenza delle ricariche.
- **Migliorare la visibilità dei droni**: Ottimizzare i sensori e le tecnologie di visione per una sorveglianza più efficace.
- **Ridurre l'area da sorvegliare**: Concentrarsi su aree di maggiore importanza per una copertura più efficiente.
- **Ridurre il tempo di ricarica**: Implementare stazioni di ricarica rapida per minimizzare i tempi di inattività dei droni.
- **Aumentare l'intervallo per la sorveglianza di ogni punto**

In conclusione, nonostante le sfide affrontate, il progetto ha dimostrato che è possibile gestire con successo un sistema complesso di droni attraverso una comunicazione e un coordinamento efficaci.