# **Лабораторні Роботи**

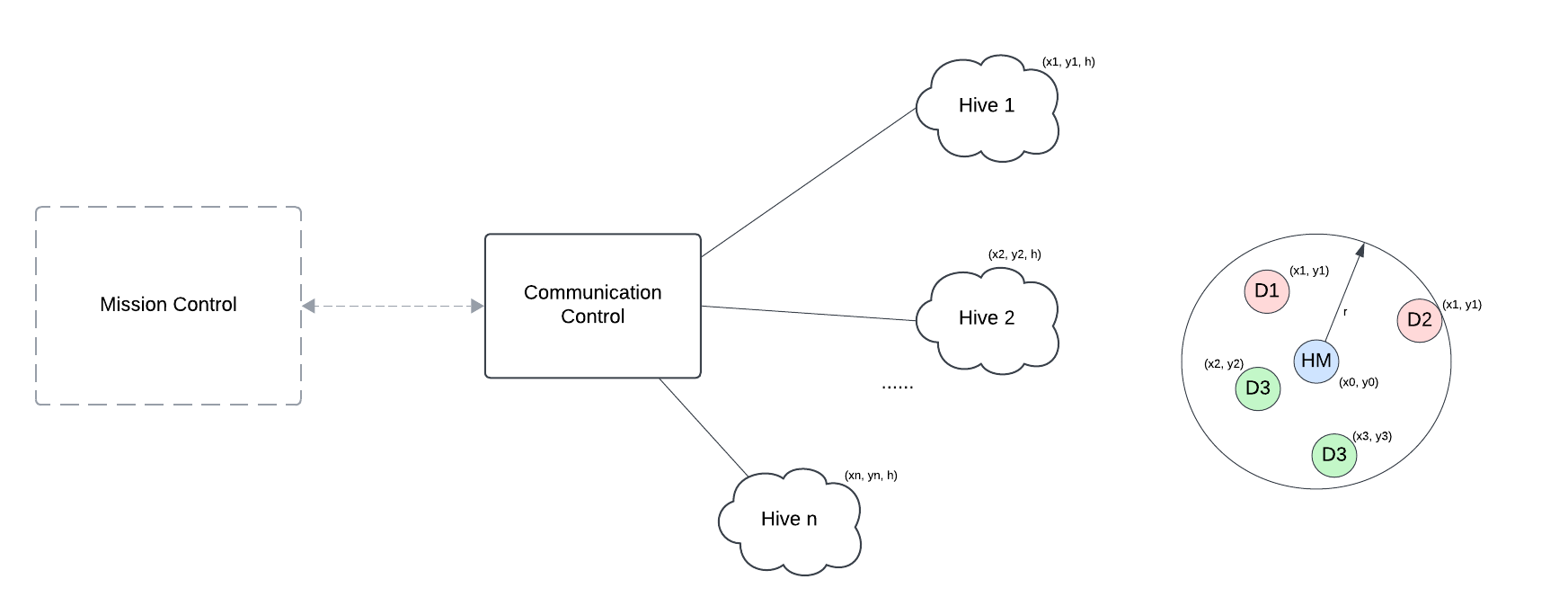
## Перелік лабораторних робіт

Кожна лабораторна робота має включати компоненту діаграму, діаграму розгортання і Flow Diagram (для CI/CD, IaC і моніторингу)

1. Розгортання застосунку за допомогою IaaC підходу.
   * Розгорнути VPS в обраній хмарі (приклади + локально)
   * Розгорнути застосунок без використання CI/CD підходу
   * **\*** Налаштувати Linux Firewall на сервері
   * **\*** Налаштувати балансувальник навантаження (Load Balancer) - локально чи за допомогою
2. Розгортання застосунку за допомогою PaaS підходу в обраній хмарі
   * Розгорнути застосунок PaaS підходу (наприклад: Azure AppService, AWS Elastic Beanstalk, Heroku, Digital Ocean App Platform)
   * Проаналізувати різницю розгортання з IaaC
   * **\*** Налаштувати автоматичне масштабування
3. Розгортання застосунку за допомогою Docker Container
   * Створити Docker Container для застосунку
   * Розгорнути застосунок локально чи на VPS
4. Розгорнути застосунок за допомогою Docker Compose
   * Розмістити Docker Container в Container Registry
   * Розгорнути застосунок і всі допоміжні сервіси за допомогою docker-compose
   * Пояснити налаштування docker-compose (мережа, volumes, залежність між компонентами)
5. Розгортання застосунку за допомогою CaaS (Kubernetes)
   * Використати Docker Container з лабораторної роботи 3
   * Розмістити Docker Container в Container Registry хамарного провайдера (наприклад Azure Container Registry, Digital Ocean Container Registry, AWS ECR)
   * Розгорнути застосунок в Kubernetes в хмарі (наприклад AKS, EKS, Digital Ocean Kubernetes) або локально за допомогою mini-kube
6. Налаштування CI/CD процесу
   * Створити pipeline для застосунку: тестування, компіляції, створення Docker Container, копіювання контейнера в Container Registry
   * Автоматично розгорнути застосунок в VPS або Kubernetes з pipeline
   * **\*** Підтримати Git Flow - дозволити розгортати з DEV і MAIN гілок на різні сервери
   * **\*** Підтримати автоматичний та ручний запуск операцій
   * **\*** Реалізувати статичне тестування коду (SAST) за допомогою безкоштовних сервісів
7. \* Налаштування системи моніторингу для застосунку (РГР)
   * Зробити порівняльний аналіз варіантів моніторингу для вашого випадку (врахувати вартість, складність розгортання і складність підтримування)
   * Розгорнути систему моніторингу або використати безкоштовне SaaS рішення (наприклад ELK або Grafana + Loki + Prometheus або Azure Monitor або Zabbix)
   * Реалізувати збір логів з застосунку з можливістю фільтрації по типу запису і часу
   * **\*** Створити Monitoring Dashboard який би давав чітке розуміння стану системи
   * **\*** Налаштувати систему сповіщення про інциденти
   * **\*** Налаштувати систему моніторингу інфрастуктури (VPS або Kubernetes) яка б збирала і відображала низькорівневі метрики

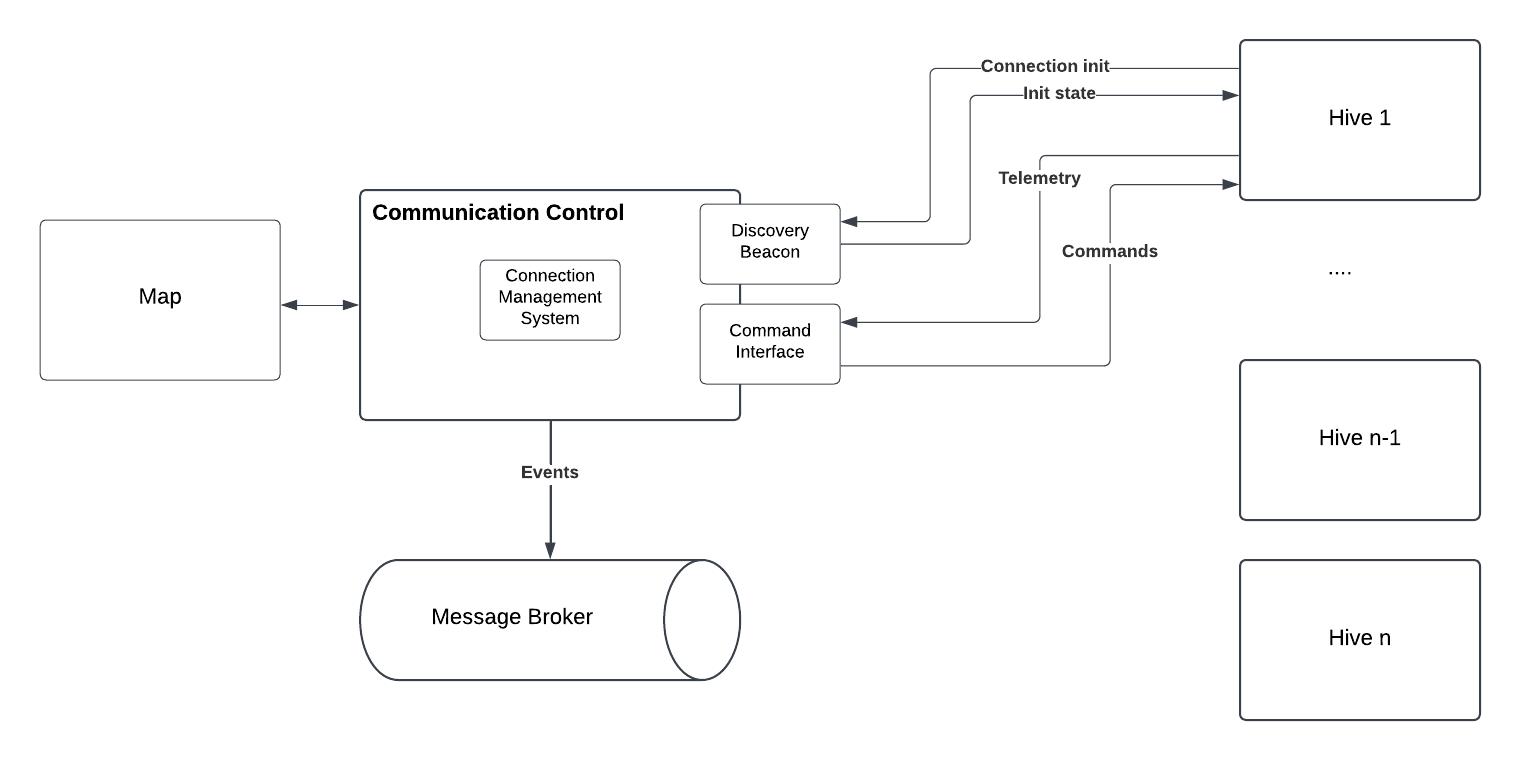
## Функціональні вимоги застосунку

Застосунок призначений для моделювання одночасної взаємодії з кількома роями дронів, що передбачає підключення до дронів, відображення їх місцеположення на інтерактивній мапі та передачу команд управління. З метою оптимізації структури мережі дронів, вона моделюється через один головний дрон (мастер-дрон).

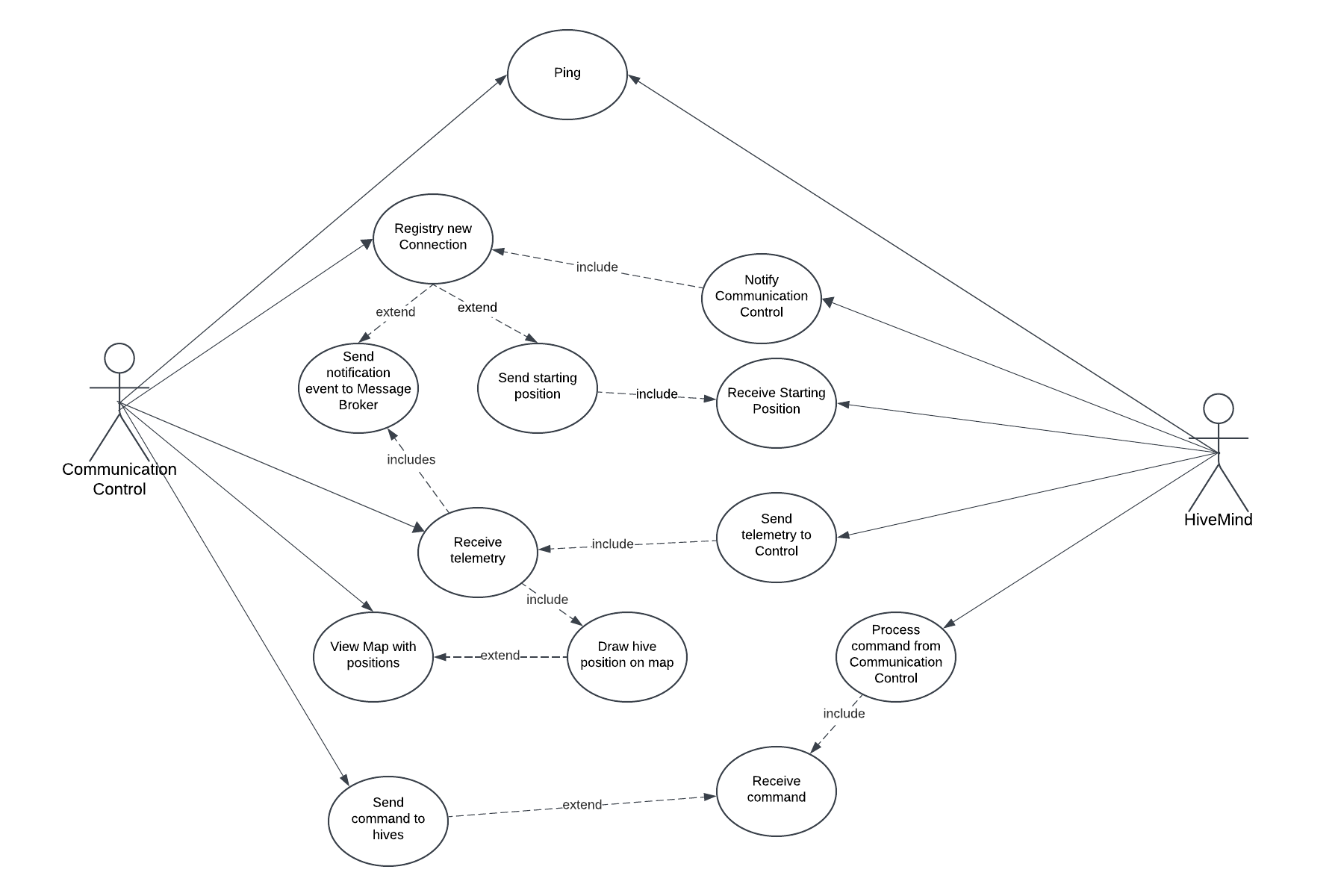


Архітектура рішення складається з двох основних підсистем:

* **Центр комунікації** — включає систему управління комунікацією, модуль візуалізації на мапі та шину даних для отримання подій із системи.
* **Емулятор рою дронів** (Hive 1, ... n) — відповідає за симуляцію роботи кожного рою.



Частини системи мають комунікувати через HTTPS протокол (додатково за бажанням може бути розроблена комунікація по gRPC протоколу)



### Communication Control

| ID | Вимога | Коментар |
| --- | --- | --- |
| СС-1 | Центр керування виставляє endpoint для реєстрації підключення | * Hive IP/Host + Port * Hive version * Hive ID |
| СС-2 | Інформація про підключення має зберігатися в пам’яті разом з останніми параметрами телеметрії | * IP, Port * Hive ID * Назначений колір |
| СС-3 | Центр керування у відповідь на підключення висилає початковий стан рою | * Координати області полігону (GPS) * Початкові координати рою (GPS) * Початкова висота рою * Швидкість рою * Інтервал оновлення телеметрії * Ping-інтервал |
| СС-4 | Центр керування надає endpoint для отримання телеметрії від рою, яка висилається з заданим інтервалом | * Поточна GPS координата в рамках полігону * Швидкість * Висота * Стан (стоп, переміщення) |
| СС-5 | Центр керування висилає команду обраному рою на IP/Port які були задані при підключенні | Команди:   * Стоп * Перемістись в координати * Надай телеметрію |
| СС-6 | Центр керування виводить на мапу всі підключенні рої (остання відома GPS координата). Колір відображення задаєтся при підключенні | Для реалізації мапи можна використати  <https://openlayers.org/>  або  <https://www.openstreetmap.org> |
| CC-7 | Якщо дані GPS роя не оновлювались більше 30 секунд, на мапі рій помічається сірим |  |
| СС-8 | Центр керування має надавати endpoint для реалізації Ping механізму - рій буде посилати запит на перевірку підключення з заданим інтервалом |  |
| CC-9 | Центр керування має розсилати існуючим підключенням Ping запит з заданим інтервалом. Якщо рій не відповідає впродовж 10 PING запитів, то він видається з центру керування |  |
| СС-10 | При отриманні підключення чи телеметрії центр комунікацій має відправити цю інформацію в Шину Даних |  |
| CC-11 | Центр комунікацій має логувати час виконання запиту до рою у вигляді APM метріки |  |

Події які мають слатись в шину даних центром комунікації:

* Приєднання нового рою
* Рій відключився
* Рій надіслав телеметрію
* Статус рою змінено

### Hive Mind

| ID | Вимога | Коментар |
| --- | --- | --- |
| HM-1 | HiveMind при старті має ініціювати підключення до центру керування відправивши запит на підключення | Якщо не вдалось встановити підключення до центру керування, то система має продовжувати відправляти запити з певним інтервалом допоки підключення не буде встановлене |
| HM-2 | У відповідь на ініціалізацію підключення, центр комунікацій поверне початковий стан системи. Стан має бути збережений в системі і використовуватись для подальших розрахунків | * Координати області полігону (GPS) * Початкові координати рою (GPS) * Початкова висота рою * Швидкість рою * Інтервал оновлення телеметрії * Ping-інтервал |
| HM-3 | HiveMind має відправляти поточний стан системи з заданим інтервалом на центр керування | * Поточна GPS координата в рамках полігону * Швидкість * Висота * Стан (Стоп, Переміщення, Помилка) |
| HM-4 | HiveMind має надавати endpoint для отримання команд від центру керування | * Стоп * Перемістись в координати * Надай телеметрію |
| HM-5 | При отриманні команди “Переміститись в координати” система має почати емуляцію руху з заданою швидкістю. Стан системи змінюється на “Рух” |  |
| HM-6 | При отриманні команди “Стоп” система має припинити рух і перейти в стан “Стоп” |  |
| HM-7 | При отриманні команди “надати телеметрію” система відправляє поточний стан системи (див. HM-3) |  |
| HM-8 | HiveMind має надавати endpoint для реалізації Ping механізму. Центр комунікації буде посилати запит на перевірку підключення |  |
| HM-9 | HiveMind має посилати Ping запит на центр керування з заданим інтервалом. Якщо центр комунікації не відповідає впродовж 10 запитів, то рій переходить в стан “Помилка” і намагаєтся ініціювати нове підключення |  |

## Нефункціональни вимоги застосунку

* Система має реалізовувати систему логування (в файл або stdout) з різними рівнями логування (INFO, DEBUG, ERROR, тощо). Додатковим плюсом буде реалізація системи логування і APM за допомогою [Open Telemetry протоколу](https://opentelemetry.io/).
* Комунікація має бути реалізована за допомогою протоколу HTTPS (додатково можна реалізувати протокол gRPC)
* Додаткові бали будуть нараховані у випадку реалізації HiveMind на GoLang, C++ або Rust

## Протокол Комунікації

### Ping

**Communication Control: GET /ping**

**HiveMind: GET /ping**

Request Payload

| {  "timestamp": "DateTime",  "hiveID": string } |
| --- |

Response Payload

| {  "status": "OK",  "timestamp": DateTime } |
| --- |

### Ініціалізація підключення

**Communication Control: POST /api/v1/connect**

Request Payload

| {  "hiveIP": String,  "hivePort": Int,  "hiveID": String, } |
| --- |

| Поле | Тип | Коментар |
| --- | --- | --- |
| hiveIP | String | IP за яким доступний API HiveMind |
| hivePort | INT | Порт за яким доступний API |
| hiveID | String | Унікальний ідентифікатор рою. Задаєтся окремо від системи |

Response Payload

| {  "area": "GPS-Polygon",  "initLocation":{  "Lat": “float”,  "Long": “float”  },  "initHeigh": "float",  "speed": "float",  "telemetryIntervalMs": "int",  "pingIntervalMs": "int", } |
| --- |

| Поле | Тип | Коментар |
| --- | --- | --- |
| area | GPS Area | Полігон |
| initLocation | {“Lat”: float, “Long”: float} | Початкові координати рою |
| initHeigh | String | Унікальний ідентифікатор рою. Задаєтся окремо від системи |
| speed | float | Швидкість з якою має змінюватись координата |
| telemetryIntervalMs | int | Інтервал з яким рой має надсилати поточний стан |
| pingIntervalMs | int | Інтервал з яким рой має надсилати PING запит для перевірки підключення |

### Відправка телеметрії

**Communication Control: POST /api/v1/telemetry**

Request Payload

| {  "hiveID": string  "location": {"Lat": float, "Long": float},  "speed": float,  "height": float,  "state": enum } |
| --- |

| Поле | Тип | Коментар |
| --- | --- | --- |
| location | {“Lat”: float, “Long”: float} | Поточна координата |
| hiveId | string | Унікальний ідентифіктаор рою |
| speed | float | Поточна швидкість |
| height | float | Поточна висота |
| state | enum:   * Stop * Move * Error | Поточний стан |

Response Payload

| {  "status": "OK"  "timestamp": DateTime } |
| --- |

### Відправка команди

**HiveMind: POST /api/v1/command**

Request Payload

| {  "commandType": enum,  "commandPayload": {<dynamic object>},  "timestamp": Datetime } |
| --- |

| Поле | Тип | Коментар |
| --- | --- | --- |
| commandType | enum:   * Stop * Move * GetTelemetry | Тип команди |
| commandPayload | Dynamic object | Визначаєтся в залежності від типу команди  Stop/GetTelemetry: null  Move:  {  Lat: float,  Lon: float,  Height: float  } |
| timestamp | DateTime | Поточна дата команди |

Response Payload

У відповідь HiveMind відправляє свою телеметрію