УДК: 629.737.5:519.87 DOI: 10.54858/dndia.2024-20-5

Л.М. Артюшин[1], О.А. Кононов[1], А.Ю. Кир'янов[2]

[1] *Державний науково-дослідний інститут авіації, Київ*
[2] *Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського", Київ*

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КЕРУВАННЯ ГРУПОЮ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ

У статті розглядаються ключові підходи до створення архітектури програмного забезпечення для ефективного керування групою безпілотних літальних апаратів (БпЛА) в умовах сучасних викликів і технологічних можливостей. Описуються методи координації дій небагатьох літальних апаратів, а також використання алгоритмів для забезпечення оптимального обміну інформацією між ними. Розглянуто питання безпечного та надійного зв'язку між БпЛА та наземним центром керування, а також способи обробки даних у режимі реального часу для підтримки автономності та гнучкості систем. Стаття окреслює перспективи та основні проблеми для побудови програмного забезпечення, яке дозволяє ефективно керувати групою БпЛА, включно з можливостями інтеграції з існуючими технологіями.

Ключові слова: *безпілотна авіація, спільна авіаційна група, алгоритми прийняття рішень, коефіцієнт ефективності управління, розподілені обчислення та зберігання даних, обробка даних у реальному часі.*

Вступ

Постановка проблеми. Розробка ефективної архітектури програмного забезпечення для керування групою безпілотних літальних апаратів (БпЛА) є складним завданням, яке потребує вирішення низки технологічних і операційних проблем. У сучасному світі зростає потреба в автономних системах, здатних виконувати комплексні завдання у віддалених або небезпечних умовах, включаючи збройні конфлікти, моніторинг навколишнього середовища, пошуково-рятувальні операції та інше [1, 2]. Однак керування групою БпЛА висуває вимоги до програмної архітектури, що має забезпечити високу продуктивність, стійкість до збоїв та безпечний обмін даними.

Головні виклики полягають у координації великої кількості БпЛА в режимі реального часу, що вимагає високої надійності зв'язку між ними, обробки великого обсягу даних і швидкого прийняття рішень. Система повинна мати можливість розподіленого обчислення, щоб уникнути критичного зв'язку з центральним вузлом, і підтримувати адаптивність у випадках зміни навколишнього середовища або втрати зв'язку. Незалежно від цього, зростання масштабів БпЛА потребує архітектури, яка є легко згрупованою і захищеною від загрози кібербезпеки.

Таким чином, важливим є завдання розробити архітектуру програмного забезпечення, яка відповідала б цим вимогам, забезпечуючи надійне і безпечне керування групою БпЛА з можливістю масштабування.

Аналіз останніх досліджень і публікацій. У дослідженнях, присвячених архітектурі програмного забезпечення для керування групою БпЛА, включно розглядаються питання автономності, координації, безпеки та надійності системи зв'язку між БпЛА. Сучасні підходи зосереджуються на розробці адаптивних архітектур, які можуть масштабуватися відповідно до кількості БпЛА у групах та умовах.

Одним із ключових напрямів є використання розподілених систем керування, які підвищують стійкість системи до збоїв і знижують залежність від центрального вузла. Так, у роботах [3, 4] пропонується розподілити архітектуру для координаційної групи БпЛА, яка забезпечує високу швидкість обробки даних та адаптивність до змінених умов. Результати дослідження демонструють, що розподілена архітектура значно зменшує затримку в обміні інформації та зменшує завантаження в мережі, дозволяючи БпЛА швидко реагувати на нові загрози.

Інший важливий аспект — це питання безпечного та ефективного обміну даними між БпЛА. У роботі [5] досліджуються протоколи V2V (Vehicle-to-Vehicle) для зв'язку між БпЛА, що дозволяє забезпечити стабільний зв'язок між усіма

групами учасників, навіть за умов прямого зв'язку з наземним центром керування. . Такий підхід знижує ризик збоїв у критичних ситуаціях і дозволяє реалізувати комплексні операції.

Дослідники також приділяють увагу забезпеченню кібербезпеки під час обміну інформацією в групі БпЛА. У статті [6] пропонується модель захисту даних на основі шифрування й аутентифікації для забезпечення захисту комунікацій у розподілених системах керування. Підхід дозволяє значно знизити ризик несанкціонованого доступу до мережі БпЛА та підвищує стійкість системи до атаки.

Мета статті - опис розробленої архітектурної моделі програмного забезпечення системи управління та візуалізації групового польоту БпЛА, що дозволяє ефективно виконувати складні місії в реальному часі. Стаття зосереджується на інтеграції ключових модулів, включаючи базу даних, систему керування, моніторинг модуля та виявлення цілей, а також алгоритми групового управління та прийняття рішень, які забезпечують високу гнучкість, масштабованість та адаптивність системи до різних оперативних сценаріїв. Архітектура, запропонована в статті, має забезпечити надійне та безпечне функціонування групи БпЛА, гарантуючи безперервний зв'язок і стабільність виконання поставлених задач перед групою БпЛА.

Виклад основного матеріалу

Архітектурна модель програмного забезпечення. Архітектурна модель програмного забезпечення для системи управління та візуалізації групового польоту БпЛА базується на трьох основних компонентах: базі даних, яка зберігає всю необхідну інформацію про БпЛА, місії та групу; фронтенді, що показує користувачам графічний інтерфейс для дистанційного управління. Ця модель підтримує високий рівень інтеграції та взаємодії між компонентами, що дозволяє ефективно виконувати складні місії у реальному часі. Така структура сприяє гнучкому масштабуванню та адаптації системи до різних оперативних сценаріїв і вимог користувачів.

Архітектурна модель програмного забезпечення для керування групою БпЛА зображена на рис. 1.

Виходячи зі складу архітектурної моделі та загальних компонентів системи управління БпЛА, нижче наведено опис кожного модуля.

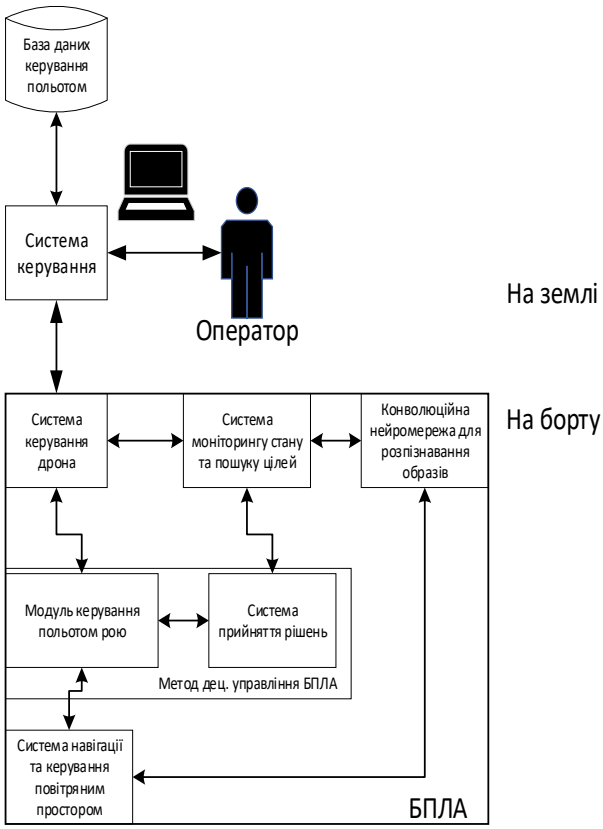

Рис. 1. Архітектурна модель програмного забезпечення для керування групою БпЛА
Джерело: розроблено авторами

1. База даних керування польотом: керує та організовує дані, необхідні для роботи БпЛА. Сюди входять журнали польотів, записи про технічне обслуговування, дані про місії та будь-яка інша відповідна інформація. Слід зазначити, що цей модуль включає також і відповідну систему управління базою даних.

2. Система керування: центральний вузол роботи БпЛА, який обробляє вхідні дані від оператора і перетворює їх на команди, що виконуються апаратом.

3. Оператор: людина-оператор, яка взаємодіє з системою БпЛА, приймаючи рішення, вводячи команди і отримуючи зворотний зв'язок від системи.

4. Система керування БпЛА: підсистема, призначена для безпосереднього управління літальним апаратом, що відповідає за виконання польотних маневрів і завдань місії відповідно до вказівок системи управління.

5. Система моніторингу та пошуку цілей: ця підсистема обробляє дані з датчиків для моніторингу стану літального апарату та навколишнього середовища, а також для ідентифікації та відстеження цілей.

6. Конволюційна нейронна мережа для розпізнавання зображень: бортовий модуль, який використовує алгоритми глибокого навчання для

обробки та інтерпретації візуальних даних для таких завдань, як виявлення, класифікація та відстеження об'єктів.

7. Модуль керування польотом групи: координує діяльність декількох апаратів у групі, забезпечуючи їхню ефективну спільну роботу для досягнення спільних цілей.

8. Система прийняття рішень: аналізує дані з усіх джерел для прийняття рішень в режимі реального часу щодо функціонування БпЛА та його реакції на фактори навколишнього середовища.

9. Система навігації та керування повітряним простором: задає рух БпЛА групи в просторі, включаючи планування траєкторії, уникнення перешкод і стабілізацію положення.

Кожен з цих модулів має вирішальне значення для ефективної та результативної роботи БпЛА, особливо коли вони використовуються в складних і динамічних умовах. Таким чином, запропонована архітектурна модель програмного забезпечення створена для координації дій БпЛА з метою безпечного та успішного виконання поставленого завдання.

Програмний код. Вимоги ефективності реалізації програмного забезпечення, його стійкості, масштабованості визначають доцільність використання Python в якості базової мови програмування.

Можливості програмування з динамічною типізацією, підтримка як об'єктно-орієнтованого, так і процедурного програмування дозволяють розробникам на Python реалізовувати найбільш підходящий підхід для вирішення конкретних задач.

Для реалізації запропонованої архітектурної моделі програмного забезпечення для керування групою БпЛА важливим є здатність Python до динамічній типізації змінних. Змінні в Python не вимагають експліцитного оголошення для визначення їх типу даних. Тип визначається автоматично під час виконання програми, що робить код більш гнучким та компактним. Це контрастує зі строгою типізацією в Java, де кожна змінна має бути оголошена з певним типом даних.

Python також відомий своїми високорівневими структурами даних, такими як списки, словники та множини, які дозволяють розробникам ефективно організовувати та маніпулювати даними. Крім того, Python має велику стандартну бібліотеку, яка включає утиліти для різноманітних завдань, починаючи від роботи з файлами та закінчуючи мережевими запитами, що значно спрощує процес розробки.

Структура розробленого програмного забезпечення показана на рис. 2.

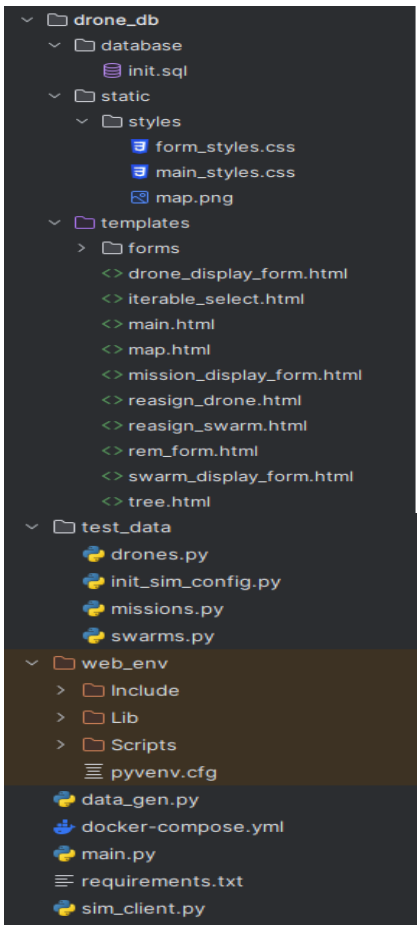

Рис. 2. Структура розробленого програмного забезпечення

Джерело: розроблено авторами

Запропонована архітектурна модель програмного забезпечення для керування групою БпЛА реалізується такими основними програмними модулями.

Файл drones.py:

цей клас перетворює дані позиції БпЛА у формат, придатний для графічного відображення. Функція convert_drone_point_to_gl_point приймає параметри, такі як позиція апарата, кольори фону та переднього плану, центр огляду та відстані поля огляду. Вона обчислює координати верхнього лівого, верхнього правого та верхнього центрального точок відображення апарата, а також координати для трикутника статусу, який вказує його стан. Функція є комплексною та потужною: можливо використовувати різні кольори для фону та переднього плану різних частин відображення БпЛА. Великий обсяг додаткової інформації можливо застосовувати для покращення інтерфейсу оператора та якості подання інформації про функціонування як окремих апаратів, так і групи у цілому.

Клас drone_control_class призначений для керування даними про БпЛА в базі даних та

забезпечення їх актуалізації. Розглядається зберігання даних у вигляді реляційної бази даних.

Методи класу:

Метод update_pos оновлює позицію БпЛА в базі даних.

Метод add додає відомості про новий БпЛА до бази даних.

Метод assign_to призначає БпЛА до конкретного групового утворення.

Метод remove видаляє БпЛА із складу групи та здійснює відповідні зміни у записах бази даних.

Метод get_rect_data повертає дані про просторове розташування апаратів, для адекватного графічного відображення процесу застосування групи. Він дозволяє керувати даними про БпЛА в базі даних із зручним інтерфейсом, а також забезпечує їх актуальність та цілісність.

Файл swarms.py:

клас swarm_control_class призначений для керування даними про групове керування в базі даних та забезпечення їх актуалізації.

Методи класу:

метод assign_to призначає групове керування в процесі виконання певної місії.

Метод add додає новий канал групового керування до бази даних.

Метод update оновлює дані про групове управління в базі даних.

Метод remove видаляє групове управління з бази даних.

Метод get_rect_data повертає дані про межи просторового розташування БпЛА, який обмежує межі групового управління на електронній мапі.

Також у коді є функція ranomize_start_swarms, яка відповідає за визначення ведучого апарата у групі.

Файл missions.py:

містить класи та функції для управління місіями окремих БпЛА групи. Клас mission_control_class включає методи для додавання, оновлення та видалення місій, а також для отримання даних про просторове положення апаратів групи, що обмежують межі місій на мапі.

Функція randomize_missions_start_data відповідає за синхронізацію у часі початку та завершення місій окремих апаратів групи.

Також у програмне забезпечення включає функції для конвертації інформації про місії БпЛА у формат, придатний для візуалізації на електронній мапі.

Цей файл є одним із основних у системі, тому розглянемо детальніше його методи.

Метод init є конструктором класу, він приймає параметри data, cursor та db_connection. Також він ініціалізує атрибути класу, такі як база даних, курсор та дані про місію, розраховує координати початкової та кінцевої точок місії, встановлює час, що залишився до закінчення дії місії.

Метод add додає новий запис про місію до бази даних з використанням параметрів, які були передані під час створення об'єкта класу. Здійснює зміни записів у базі даних.

Метод udpate оновлює існуючий запис про місію в базі даних з використанням параметрів, які були передані під час створення об'єкта класу, узгоджує внесені зміни у записах бази даних.

Метод get_rect_data отримує дані про просторове розташування апаратів для відображення на електронній мапі. Призначений для виконання запитів до бази даних, щоб отримати мінімальні та максимальні значення координат проміжних точок маршруту та координат БпЛА, які складають групу. Повертає кутові координати та координати центру мас апаратів.

Метод remove видаляє запис про місію з бази даних на основі переданих запиті, підтверджує внесення відповідних змін у базі даних.

Метод count_m_dots_array реалізує обчислення та відображення проміжних точок маршруту апаратів на мапі, ініціюючи порожній список результатів, перебираючи кожну місію для обчислення її геометричних точок початку, завершення та орієнтації, а потім додає отримані координати до цього списку.

Файл main.py:

є головним файлом програмного забезпечення. У ньому реалізовано веб-додаток на основі фреймворку Flask для управління БпЛА та їх місіями. В ньому ініціюється створення об'єктів Flask-додатку, оголошуються глобальні змінні та маршрути для обробки запитів веб-користувачів. Функція render_home керує побудовою та відображенням домашньої веб-сторінки, адаптуючи її вміст залежно від дій користувача, таких як вибір, зв'язування або видалення БпЛА зі складу групи, групових управлінь або місій окремих апаратів. Ця функція також відповідає за оновлення інформації, яка відображається на мапі, відстеження вибраних об'єктів та їх положення. Маршрути у файлі визначають, як додаток реагує на різні запити веб-сторінки, включаючи додавання нових об'єктів та управління існуючими. Також вона дозволяє користувачам оперативно керувати даними про БпЛА, групи та місії окремих апаратів через веб-інтерфейс.

Файл data_gen.py:

призначений для зчитування даних про параметри руху БпЛА, сигнали групового управління та стан місії з реляційної бази даних за допомогою бібліотеки MySQLdb. Спочатку він підключається до бази даних, потім використовує спеціальні функції для запиту інформації та

статусів об'єктів. Отримані дані організовуються у словники, де кожен об'єкт має свій унікальний ідентифікатор. В результаті, код збирає всю необхідну інформацію для подальшого використання в додатку.

Файл sim_client.py:

призначений для формування керування окремими БпЛА у групі. Вихідною інформацією є фазові координати апаратів, які отримуються із відповідних записів реляційної бази даних. Файл містить умовний цикл визначення сингалів керування для кожного БпЛА у групі відповідно до їх просторового положення, кутових координат та заданої місії для групи. Об'єктно-орієнтований принцип побудови програми дозволяє реалізувати різні методи пошуку оптимального керування БпЛА в групі без необхідності внесення кардинальних змін в код модулів програми.

Опис графічного інтерфейсу та взаємодії з ним. Загальний формат інтерфейсу є мінімалістським для зменшення інформаційного навантаження на оператора (рис.3).

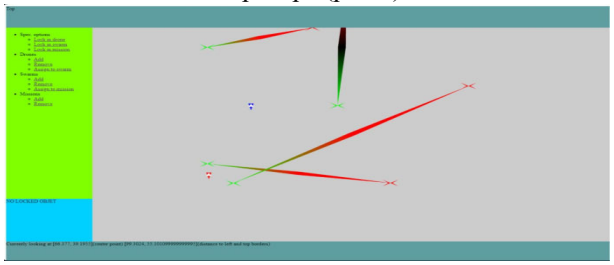

Рис. 3. Загальний вигляд інтерфейсу
Джерело: розроблено авторами

Опис окремих частин інтерфейсу оператора.

1) Формат меню вибору об'єкта для стеження показано на рис.4.

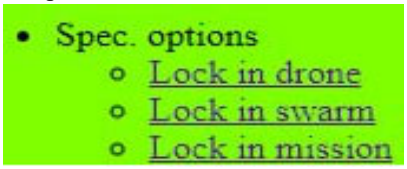

Рис. 4. Вибір об'єкта для стеження
Джерело: розроблено авторами

2) Формат параметричного налаштування графічного відображення інформації про БпЛА наведено на рис. 5.

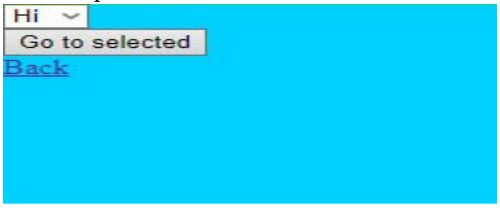

Рис. 5. Формат параметричного налаштування графічного відображення інформації
Джерело: розроблено авторами

3) Пункти меню оператора щодо визначення складу групи наведені у рис. 6.

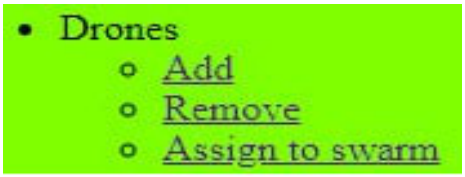

Рис. 6. Пункти меню оператора щодо визначення складу групи
Джерело: розроблено авторами

4) Формат меню для видалення БпЛА зі складу групи наведено на рис. 7.

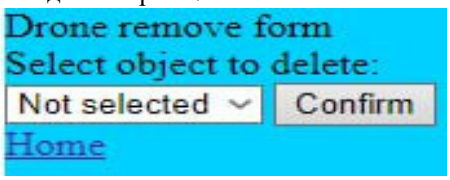

Рис. 7. Формат меню для видалення БпЛА зі складу групи
Джерело: розроблено авторами

5) Форма визначення складу групи (рис. 8).

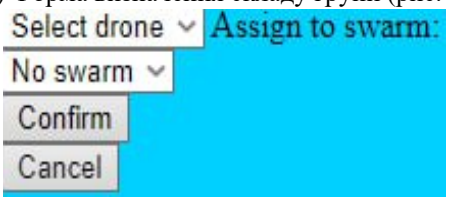

Рис. 8. Форма визначення складу групи
Джерело: розроблено авторами

6) Приклад форми для відстеження об'єкта, наведений на рис. 9.

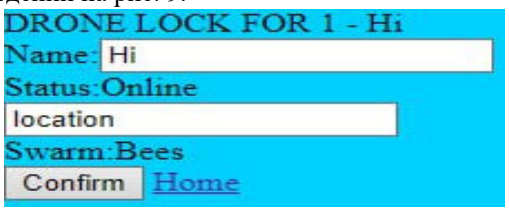

Рис. 9. Приклад форми для відстеження об'єкта
Джерело: розроблено авторами

7) Формат нижньої панелі меню інтерфейсу оператора (рис.10).

Рис. 10. Формат нижньої панелі меню інтерфейсу
Джерело: розроблено авторами

Приклади подання інформації про рух БпЛА групи. Приклади подання інформації про рух БпЛА групи в запропонованому інтерфейсі оператора представлені на рис. 11 і рис. 12, а про рух БпЛА в режимі стеження показані на рис. 13 і рис. 14.. Головним критерієм побудови інтерфейсу було намагання максимально зменшити "інформаційний тиск" на оператора.

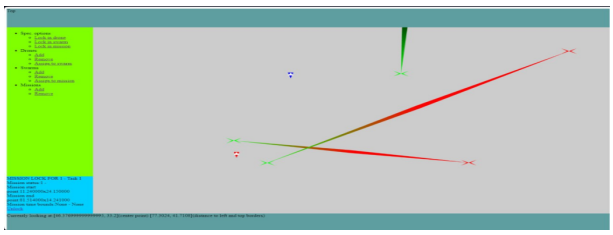

Рис. 11. Приклад подання інформації про рух БпЛА групи в запропонованому інтерфейсі оператора
Джерело: розроблено авторами

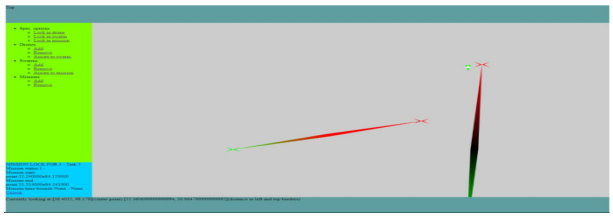

Рис. 12. Приклад подання інформації про рух БпЛА групи в запропонованому інтерфейсі оператора
Джерело: розроблено авторами

Приклади подання інформації про рух БпЛА в режимі стеження показані на рис. 13 і рис. 14.

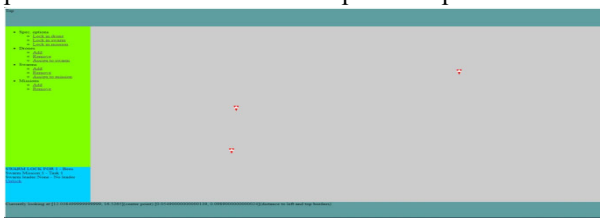

Рис. 13. Приклад стеження за груповим рухом
Джерело: розроблено авторами

Рис. 14. Приклад стеження за груповим рухом
Джерело: розроблено авторами

Практичні рекомендації щодо застосування розробленого програмного забезпечення. Технологія WebGL, яка дозволяє створювати і відображати 3D та 2D графіку прямо у веб-браузері без необхідності встановлення додаткових плагінів є зручним засобом візуалізації процесу застосування групи. WebGL робить можливим відтворення складної графіки без великого навантаження на центральний процесор, забезпечуючи плавність і високу швидкість роботи графічних програм прямо у Web.

Реалізація візуальної частини програмного забезпечення використовує модуль canvas для активації функціоналу WebGL. Якщо браузер користувача не підтримує вимоги технології WebGL, коректне відображення графіки на ньому є неможливим.

Завантаження текстур, ініціалізація нових текстур здійснюється функцією loadTexture, яка відповідає за завантаження зображень з вказаної URL. Це забезпечує наявність постійно доступної текстури, навіть якщо основне зображення ще не готове. Після завантаження зображення, воно замінює тимчасовий піксель, і, в залежності від його розмірів, для текстури встановлюються оптимальні налаштування.

Наступним етапом обробки інформації є підготовка даних для відображення просторового положення БпЛА, створюючи буфери для вершин. Вершини – це ключові точки, які використовуються для побудови геометричних форм на електронній мапі. Вони завантажуються в буфери, що дозволяє графічному процесору швидко отримувати до них доступ під час рендерингу. Буфери мають бути створені для всіх елементів сцени, включаючи БпЛА та проміжні пункти маршрутів.

Запропоноване програмне забезпечення використовує шейдерну технологію відображення інформації для оператора. Після компіляції та налаштування рейдери керують тим, як визначальні точки модифікуються та як формуються кольори на екрані. Компіляція шейдерів перед рендерингом дозволяє визначити візуальний вигляд об'єктів зони застосування групи БпЛА.

Технологія рендерингу, що реалізується для відображення інформації про групу БпЛА та стан навколишнього середовища, передбачає два етапи. На першому відображаються текстуровані об'єкти з вершинами (межами), на другому – заповнюється кольорове наповнення. Це дозволяє створити багатошарову сцену з різними візуальними ефектами. Важливість застосування саме такого підходу для відображення польотної обстановки для оператора викликана жорсткими часовими вимогами зручного представлення великого обсягу інформації.

З метою мінімізації використання обчислювальних ресурсів розроблене програмне забезпечення застосовує технологію контейнерів Docker. Замість повноцінних версій програмних модулів контейнери Docker дозволяють запускати і управляти додатками в ізольованому середовищі для зручності розробки та розгортання.

Представлена реалізація розробленого програмного забезпечення групою БпЛА вимагає застосування таких зовнішніх сервісів, які є відкритими для скачування та застосування:

- Python 3.11.4 (або пізніша, 3.11+);
- Docker Engine;
- Docker compose.

Відмова від застосування пропрієтарних (закритих корпоративних) сервісів є принциповою особливістю розробленого програмного забезпечення. Установка необхідного сервісного програмного забезпечення можлива з офіційних сайтів розробників:

Python 3.11.4 – https://www.python.org/downloads/release/python-3114/.

Docker Engine – https://docs.docker.com/desktop/install/windows-install/.

WSL (Windows Subsystem for Linux) – https://learn.microsoft.com/en-us/windows/wsl/ install або альтернативний засіб:

https://wslstorestorage.blob.core.windows.net/wslblobwsl_update_x64.msi.

Висновки

Запропонована архітектурна модель програмного забезпечення для керування групою БпЛА, що складається з реляційних баз даних, підсистеми керування, фронтенду та спеціалізованих модулів, спрямована на забезпечення ефективної інтеграції та взаємодії усіх БпЛА для виконання складних місій у реальному часі.

Її принципова відмінність від моделей програмного забезпечення для керування одиночними БпЛА полягає у необхідності введення додаткових програмних модулів, що відповідають за формування управління групою апаратів, автоматичну координацію їх руху та взаємодії з оператором групового застосування (модуль керування польотом групи, підсистема навігації та керування повітряним простором, підсистема прийняття рішень).

Запропонована архітектурна модель програмного забезпечення для керування групою БпЛА орієнтована на реалізацію в рамках як об'єктно-орієнтованого, так і процедурного програмування, спрямована на реалізацію із застосуванням відкритих сервісів. Вона надає можливість гнучкого масштабування та адаптації до різноманітних операційних сценаріїв застосування групи БпЛА, що робить її придатною для застосування в складних і динамічних умовах.

З метою забезпечення високої якості інтерфейсу оператора групи застосовано спеціальну технологію рендерингу зображень та технологію контейнерів при обробці потоків інформації.

На думку авторів, дослідження можливостей масштабування, застосування програмних технологій розподіленої обробки даних, захищеної комунікації та комбінування різних методів формування керування є доцільним **напрямом подальших досліджень** щодо вдосконалення програмного забезпечення для керування групою БпЛА.

Список літератури

1. Артюшин Л.М., Коваль В.В., Семон Б.Й., Лобанов А.А., Герасименко В.В. (2021). Підходи до формулювання стратегії управління спільними бойовими порядками пілотованої та безпілотної авіації. Науково-теоретичний та науково-практичний журнал "Наука і оборона", №4 (2021), С. 34 – 43.

2. Артюшин Л.М., Кононов О.А., Шморгун Ю.В. (2018). Базові умови прийняття рішення щодо створення автоматизованої системи управління груповим застосуванням безпілотних літальних апаратів. Сучасні інформаційні технології у сфері безпеки та оборони, № 2(35), Київ, 2018, – С. 49–54.

3. Чень, Р., Чжан, Л., Чжоу, Х. (2022). Навчання з посиленням для прийняття рішень у режимі реального часу в операціях зграї БпЛА. Штучний інтелект у робототехніці та автоматизації , 30(1), – С. 215-227.

4. Bukhvalov O., Gorodetsky V., Karsaev O, Koudryavtsev G., Samoylov V. Privacy – Preserved Distributed Coordination of Production Scheduling in B2B Networks: A Multi–agent Approach. Proc. 7th IFAC Conference on Manufacturing Modeling, Management, and Control, 2013, Volume 7. Part 1. P. 2122 – 2127.

5. Space-time continuous models of swarm robotic systems : Supporting global-to local programming / Ed. H. Hamann. Springer Science & Business Media, 2010. 160 p. DOI : 10.1007/978-3-642-13377-0.

6. Tsourdos A., White B., Shanmugavel M. Cooperative path planning of unmanned aerial vehicles. Wiley, 2011. 190 p.

7. Yershov V.V., Izvalov O.V., Nedilko S.M., Nedilko V.M. The concept of systematic control of the flight of unmanned aerial vehicles. Computer-integrated technologies: education, science, production, 2020. P. 23–30.

8. Wang X., Yadav V., Balakrishnan S. N. Cooperative UAV formation flying with obstacle/collision avoidance. IEEE transactions on control systems technology. 2007. Vol. 15, Is. 4. P. 672–679. DOI: 10.1109/TCST.2007.899191

9. Shi Z. G., Tu J., Zhang Q. et al. survey of swarm robotics system. Advances in Swarm Intelligence: Lecture Notes in Computer Science. 2012. Vol. 7331. P. 564–572.

10. Podorozhnyak A.O., Volotskov Y. A., Shevtsova O.S. Study of the control system of unmanned aerial vehicles. Adwanced information systems, 2018. P. 97–101.

11. Samborskyi I.I, Mashkov O.A., Kononov O.A. Problems of implementing joint use of promising unmanned aircraft complexes. Arsenal XXI, 2018. P. 36–41.

12. Барабаш О.В., Дахно Н.Б. Двокроковий варіаційно-градієнтний метод в системах підтримки прийняття рішень для управління безплотними літальними апаратами. Зб. наук. праць Військового інституту Київського національного університету імені Тараса Шевченка, 2017. Вип. 57. – С. 7 – 15.

13. Dakhno N., Barabash O., Shevchenko H., Leshchenko O. and Musienko A. Modified Gradient Method for K-positive Operator Models for Unmanned Aerial Vehicle Control. Proceedings of 2020 IEEE 6th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC). Conference Proceedings. October 20-23, 2020. Kyiv. P. 81 – 84.

Надійшла до редколегії 04.11.2024
Схвалена до друку 24.11.2024

Відомості про авторів:

Артюшин Леонід Михайлович

доктор технічних наук
професор
головний науковий співробітник
Державного науково-дослідного інституту авіації,
Київ, Україна
https://orcid.org/0000-0002-7488-7244

Кононов Олексій Анатолійович

доктор технічних наук
доцент
заступник начальника
Державного науково-дослідного інституту авіації,
Київ, Україна
https://orcid.org/0000-0003-2267-9109

Кир'янов Артемій Юрійович

доктор філософії
асистент кафедри
Національного технічного університету України
“Київський політехнічний інститут
імені Ігоря Сікорського”,
Київ, Україна
https://orcid.org/0000-0003-3116-0122

Information about the authors:

Leonid Artyushin

Doctor of Technical Sciences
Professor
Chief Researcher
of State Research Institute of Aviation,
Kyiv, Ukraine
https://orcid.org/0000-0002-7488-7244

Oleksii Kononov

Doctor of Technical Sciences
Assistant Professor
Deputy Chief
of State Research Institute of Aviation,
Kyiv, Ukraine
https://orcid.org/0000-0003-2267-9109

Artemii Kyrianov

Doctor of Philosophy, Ph.D
Assistant Department
of the National Technical University
of Ukraine “Igor Sikorsky
Kyiv Polytechnic Institute”,
Kyiv, Ukraine
https://orcid.org/0000-0003-3116-0122

SOFTWARE ARCHITECTURE FOR MANAGING A GROUP OF UAVS

L. Artyushin, O. Kononov, A. Kyrianov

The article explores the main approaches to creating a software architecture for controlling a group of unmanned aerial vehicles (UAVs) in the context of modern technological capabilities and challenges. The growing need for complex autonomous systems to perform various tasks, such as surveillance, monitoring and rescue operations, all require new approaches to controlling multiple drones, ensuring their coordination, safety and reliability. A method of coordinating actions between multiple drones is discussed in detail, which allows for more efficient information exchange and task performance in a group. In addition, considerable attention is paid to algorithms that ensure optimal communication between UAVs and help coordinate their activities to achieve a common goal. The article also emphasises the importance of reliable and secure communication between the UAV and the operator's ground centre, which is especially important for tasks that require a rapid response to changing environmental conditions. Real-time data processing approaches that rapidly increase the autonomy and flexibility of the system, allowing drones to adapt to changing conditions and make optimal decisions within their tasks. UAV group management software supports distributed information processing, which reduces the load on the central node and reduces the risk of malfunctions of individual drones.

In addition, the article outlines the prospects for integrating the proposed architecture with existing automated control system technologies. The main challenges, such as scaling systems, ensuring cybersecurity during data exchange, and developing algorithms to optimise the resources of a group of UAVs, are considered. This approach makes it possible to create efficient and reliable UAV group control software that can adapt to different operational scenarios and ensure safe task performance. Due to distributed data processing, secure communication, and integration of deep learning algorithms, the system maintains high autonomy and reliability during the mission. The article outlines the prospects and main challenges for building software that allows for effective management of a group of UAVs, including the possibility of integration with existing technologies.

Keywords: unmanned aviation, joint aviation group, decision-making algorithms, management efficiency coefficient, distributed computing and data storage, real-time data processing.