МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2025 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ДОДАТКУ ДЛЯ ЗНАХОДЖЕННЯ ОПТИМАЛЬНОГО ШЛЯХУ БЕЗПІЛОТНОГО ЛІТАЛЬНОГО АПАРАТУ

Виконав – студент 4 курсу, групи 41-з-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Касумов Максим Ільгамович

Науковий керівник – к.е.н., доцент
Концеба Сергій Михайлович

Умань – 2025

ЗМІСТ

ВСТУП

РОЗДІЛ 1 АНАЛІЗ РОЗВИТКУ СИСТЕМ ЗНАХОДЖЕННЯ ОПТИМАЛЬНОГО ШЛЯХУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1. Аналіз стану систем знаходження оптимального шляху безпілотного літального апарату

1.2. Порівняльний аналіз аналогів

1.3. Вплив алгоритмів знаходження оптимального шляху на ефективність безпілотного літального апарату

1.4. Постановка задачі

РОЗДІЛ 2 РОЗРОБКА МЕТОДІВ, МОДЕЛІ ТА АЛГОРИТМІВ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1. Аналіз вхідних та вихідних даних системи

2.2. Розробка загального алгоритму роботи системи та методу визначення оптимального шляху на основі алгоритмів Дейкстри та A*

2.3. Розробка методу врахування характеристик безпілотника та моделі системи

2.4. Аналіз і обґрунтування вибору мови програмування та обґрунтування вибору засобу відображення мапи

РОЗДІЛ 3 РОЗРОБКА СТРУКТУРИ ТА ПРОГРАМНИХ МОДУЛІВ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1. Розробка структури та інтерфейсу мобільного застосунку

3.2. Використання принципів об’єктно-орієнтованого програмування при розробці мобільного застосунку

3.3. Розробка модуля локального збереження даних

3.4. Тестування мобільного застосунку

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та автоматизації процесів, безпілотні літальні апарати (БПЛА) посідають все більш вагоме місце у різноманітних сферах людської діяльності. Вони активно використовуються у військовій справі, сільському господарстві, екологічному моніторингу, аерозйомці, логістиці, геодезичних вимірюваннях, нагляді за інфраструктурою, рятувальних операціях та в багатьох інших напрямках. З розвитком штучного інтелекту та систем автоматичного управління зростає потреба у створенні програмних рішень, які забезпечують автономність, адаптивність та ефективність роботи БПЛА в реальному середовищі.

Однією з найбільш критичних задач при експлуатації безпілотників є **визначення оптимального маршруту польоту**, який забезпечує досягнення цілі з мінімальними витратами ресурсів, часу та енергії, при цьому гарантуючи безпеку та уникнення небезпечних зон (наприклад, перешкод, заборонених для польоту територій, зон радіоперешкод тощо). У зв’язку з цим виникає необхідність у застосуванні математичних методів та алгоритмів маршрутизації, що враховують різноманітні параметри навколишнього середовища, а також особливості самого апарату (тип, енергоємність, швидкість, вантажопідйомність тощо).

Існуючі програмні засоби, що використовуються для управління БПЛА, зазвичай не мають повноцінної підтримки інтелектуального планування маршруту в умовах зміни рельєфу, погодних умов або появи нових перешкод. Крім того, більшість таких систем орієнтовані на настільні платформи або спеціалізоване обладнання, що обмежує їхнє використання в мобільних умовах. Розробка мобільного застосунку, який дозволяє швидко й ефективно знаходити оптимальний маршрут безпілотного літального апарату з урахуванням просторових даних, змін середовища та інших обмежень, є актуальним завданням як з наукової, так і з практичної точки зору.

Зважаючи на поширення Android-пристроїв, доцільним є створення саме мобільного застосунку, який забезпечує зручність управління БПЛА в польових умовах, можливість автономної маршрутизації, інтеграцію з картографічними

сервісами (Google Maps, OpenStreetMap), а також використання ефективних алгоритмів пошуку найкоротшого або найоптимальнішого шляху (наприклад, алгоритмів Дейкстри, A*, Theta*, RRT тощо).

Таким чином, актуальність даної теми зумовлена поєднанням практичної потреби у швидкому та безпечному визначенні траєкторії польоту безпілотників, широкими можливостями сучасних мобільних технологій і необхідністю гнучкого, адаптивного та інтелектуального підходу до планування маршрутів в умовах реального середовища.

Метою дослідження є розробка мобільного застосунку для платформи Android, що реалізує побудову оптимального маршруту польоту безпілотного літального апарату з урахуванням заданих критеріїв (найкоротша відстань, безпека, енергоефективність тощо).

Для досягнення мети було поставлено такі **завдання дослідження**:

- Провести аналіз існуючих систем та алгоритмів маршрутизації для БПЛА.
- Визначити вхідні та вихідні дані, необхідні для формування маршруту.
- Обрати та обґрунтувати ефективний алгоритм побудови шляху (наприклад, алгоритм Дейкстри, A* тощо).
- Розробити архітектуру мобільного додатку та спроектувати його інтерфейс.
- Реалізувати основні функціональні модулі застосунку на базі мови Kotlin (або Java) з використанням відповідних API (Google Maps API або OpenStreetMap).
- Провести тестування додатку в умовах наближених до реальної експлуатації.
- Проаналізувати результати функціонування застосунку та надати рекомендації щодо його подальшого розвитку.

Об’єктом дослідження є процес планування маршруту для безпілотного літального апарату в автономному або напівавтономному режимі.

Предметом дослідження є програмні методи і засоби побудови оптимального маршруту польоту БПЛА за допомогою мобільного додатку.

Методи дослідження, що використовувалися в роботі, включають:

- аналіз і систематизацію літературних джерел та прикладних систем,
- методи об’єктно-орієнтованого програмування,
- графові алгоритми (зокрема, алгоритм Дейкстри, A*),
- методи проектування та розробки програмного забезпечення,
- моделювання та тестування у віртуальних або реальних умовах.

РОЗДІЛ 1 АНАЛІЗ РОЗВИТКУ СИСТЕМ ЗНАХОДЖЕННЯ ОПТИМАЛЬНОГО ШЛЯХУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану систем знаходження оптимального шляху безпілотного літального апарату

Сучасний світ стрімко розвивається, і з кожним днем технології стають все більш інноваційними та інтегрованими у повсякденне життя. Однією з таких інновацій є безпілотні літальні апарати (БПЛА), які відкривають нові можливості для різних сфер діяльності, від сільського господарства до моніторингу та доставки [5]. Однак, ефективне використання БПЛА потребує розробки надійних систем знаходження оптимального шляху, які б забезпечували безпеку, точність та ефективність польотів.

Задача знаходження оптимального шляху є однією з ключових у процесі автономного управління безпілотними літальними апаратами (БПЛА). Від ефективності цієї системи залежить здатність апарата безпечно та швидко досягати заданої цілі, уникати перешкод, зменшувати витрати енергії та ефективно працювати у динамічному середовищі.

Існує кілька основних підходів до реалізації систем знаходження оптимального маршруту. Умовно їх можна поділити на глобальні, локальні та гібридні. Глобальні методи передбачають наявність повної карти середовища — вони обчислюють маршрут з урахуванням усієї доступної інформації. До таких методів відносять алгоритми A*, Dijkstra, RRT (Rapidly-exploring Random Tree) та генетичні алгоритми. Вони здатні знайти оптимальний або майже оптимальний маршрут, проте часто мають високу обчислювальну складність, особливо у великомасштабних просторах.

Локальні методи базуються на даних із сенсорів, які БПЛА отримує в режимі реального часу. Вони не потребують повної карти і приймають рішення безпосередньо в польоті. Прикладом є метод потенційних полів або Dynamic Window Approach. Основною перевагою цих методів є швидкість реагування, проте вони схильні до застрягання в локальних мінімумах, що унеможливлює знаходження глобально оптимального маршруту.

Гібридні системи поєднують обидва підходи: глобальне планування використовується для розрахунку основного маршруту, а локальні методи — для адаптації до змін навколишнього середовища, таких як поява несподіваних перешкод.

В останні роки спостерігається активний розвиток підходів на основі штучного інтелекту, зокрема глибинного та підкріплювального навчання. Алгоритми на кшталт Deep Q-Networks (DQN), Proximal Policy Optimization (PPO) та інші демонструють здатність до самонавчання, що дозволяє БПЛА адаптувати свою поведінку до нових умов без попереднього програмування. Крім того, у таких системах часто застосовується комбінація з технологією SLAM (Simultaneous Localization and Mapping), яка дозволяє одночасно будувати карту середовища та планувати маршрут.

Попри значні досягнення, існує ряд проблем, які обмежують впровадження автономних систем маршрутизації у складних умовах. Серед них: наявність динамічних перешкод, обмеження обчислювальних ресурсів на борту апарата, необхідність функціонування в режимі реального часу, а також вплив погодних умов та зовнішніх електромагнітних завад.

Таким чином, системи знаходження оптимального шляху для БПЛА перебувають у фазі активного дослідження та вдосконалення. Основні напрямки розвитку включають підвищення адаптивності, швидкодії та надійності алгоритмів, з особливим акцентом на використання штучного інтелекту, розподілених обчислень та хмарних сервісів [8].

1.2 Порівняльний аналіз аналогів

Використання безпілотних літальних апаратів стає все більш популярним у різних сферах. Це пов’язано з їхньою здатністю виконувати широкий спектр завдань, від моніторингу та картографування до доставки вантажів і надання допомоги в надзвичайних ситуаціях. Безпілотники використовуються в сільському господарстві, будівництві, енергетиці та багатьох інших галузях. Розробка програмних засобів системи знаходження оптимального шляху

безпілотного літального апарату може привернути увагу до проблематики оптимізації процесу пошуку найкоротшого маршруту для дрона [23].

До найбільш відомих рішень відносять:

− PathPlanner;

− PX4 Autopilot;

− Mission Planner Home;

− QGroundControl;

− DJI Gs Pro.

PathPlanner (рис. 1.1) — це програмна або апаратно-програмна система, призначена для побудови оптимального маршруту безпілотного літального апарата (БПЛА) з точки старту до цільової позиції з урахуванням наявних перешкод, обмежень навколишнього середовища та цілей місії. Основною метою PathPlanner є забезпечення безпечного, ефективного й надійного переміщення апарата в просторі, враховуючи різноманітні фактори: від рельєфу місцевості до погодних умов або зон із забороною польоту.

Робота PathPlanner зазвичай починається з отримання інформації про просторову конфігурацію середовища, яка може надходити із попередньо завантаженої карти, сенсорів на борту БПЛА або з зовнішніх джерел (наприклад, супутникових даних або наземних станцій). Після цього система аналізує отриману інформацію, визначає поточне положення БПЛА та координати цільової точки, після чого розраховує маршрут.

PathPlanner використовує певні алгоритми планування шляху. У випадку відомого середовища та статичних перешкод, зазвичай застосовуються алгоритми графового типу — наприклад, A* або Dijkstra. Вони дозволяють знайти найкоротший шлях на сітчастій або графовій моделі простору. Якщо середовище змінюється динамічно, або якщо маршрут не можна повністю розрахувати заздалегідь, застосовуються адаптивні або стохастичні методи — зокрема, метод потенційних полів, RRT (Rapidly-exploring Random Tree) або методи штучного інтелекту.

Окрім знаходження шляху, PathPlanner виконує ще низку важливих функцій. Це, зокрема, оновлення маршруту в режимі реального часу (replanning)

у разі виникнення нових перешкод, оптимізація енерговитрат (наприклад, мінімізація набору висоти або різких маневрів), а також контроль відповідності маршруту заданим параметрам безпеки.

Сучасні системи PathPlanner інтегруються з автопілотом БПЛА, модулями виявлення перешкод, а також з комунікаційними системами, що дозволяє забезпечити повну автономність апарата або ж можливість втручання оператора при потребі. Деякі реалізації PathPlanner використовують елементи машинного навчання, що дозволяє системі вдосконалювати свої маршрути з досвідом.

Таким чином, PathPlanner є одним із ключових елементів у структурі керування БПЛА. Від його ефективності безпосередньо залежить здатність літального апарата виконувати складні задачі — як у цивільних (наприклад, картографування, доставка вантажів), так і у військових застосуваннях. Постійний розвиток алгоритмів, обчислювальних потужностей та сенсорних технологій відкриває нові можливості для вдосконалення цих систем у найближчому майбутньому [9].

Інтерфейс програмного застосунку «PathPlanner» зображено на рисунку 1.1.

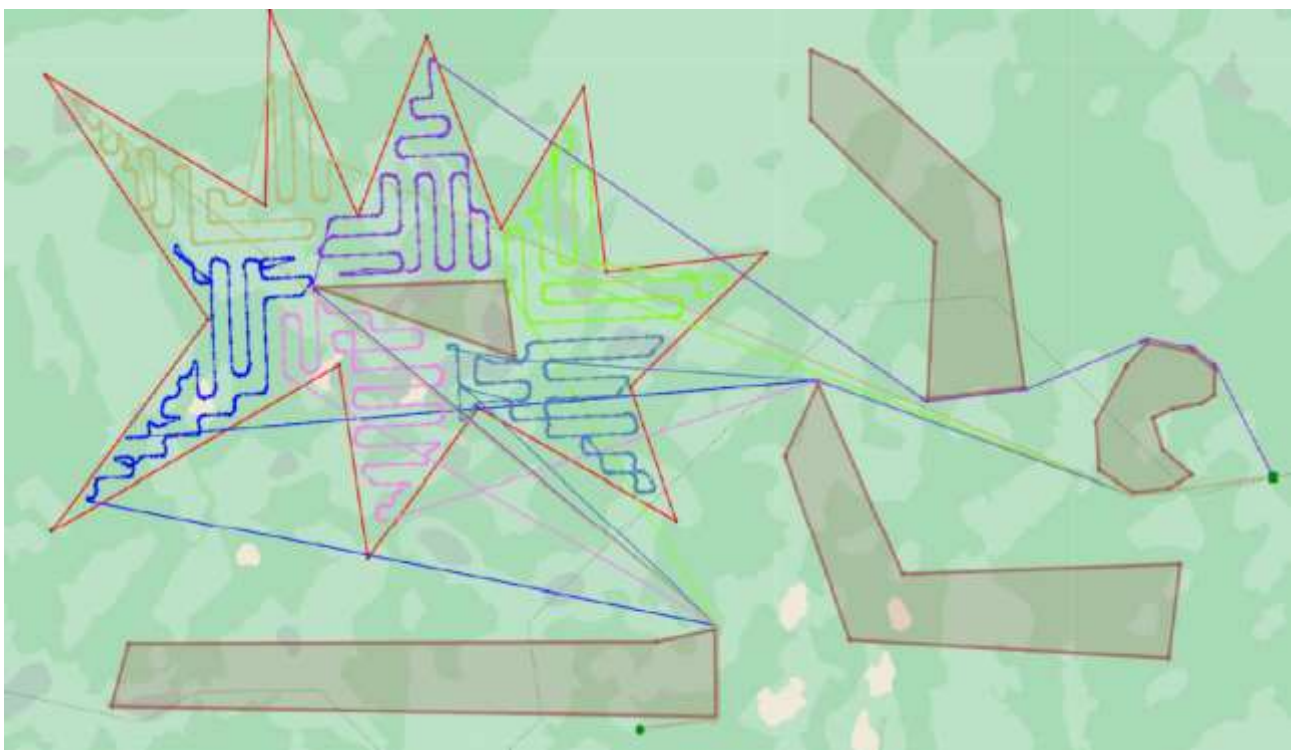


Рис. 1.1 Інтерфейс програмного застосунку «PathPlanner»

PX4 Autopilot — це високотехнологічна, модульна та відкрита платформа для автономного управління безпілотними літальними апаратами різних типів:

від мультикоптерів і літаків до наземних або підводних дронів. Вона активно розвивається міжнародною спільнотою розробників, дослідників і компаній у межах проєкту з відкритим кодом під егідою Dronecode Foundation. Завдяки гнучкій архітектурі, PX4 може адаптуватися під найрізноманітніші потреби — як комерційні, так і наукові чи освітні.

PX4 складається з ядра автопілота, системи управління польотом (Flight Control System), набору алгоритмів стабілізації, навігації та планування маршруту, а також комунікаційних інтерфейсів для взаємодії з датчиками, наземними станціями, камерами, GPS, системами уникнення перешкод тощо. Важливою особливістю цієї системи є те, що вона підтримує роботу як у повністю автономному режимі, так і з частковим втручанням оператора.

Архітектура PX4 базується на операційній системі реального часу NuttX або, у новіших версіях, на вільній операційній системі Linux для складніших завдань. Вона включає окремі модулі для керування положенням у просторі (позиціонування), стабілізації, планування місій, виявлення відхилень у роботі, відновлення після збоїв тощо. Такий модульний підхід дозволяє користувачу налаштувати систему відповідно до потреб місії — від простих польотів по GPS до складних сценаріїв, де потрібно взаємодіяти з мережею інших дронів або працювати в складному середовищі з перешкодами.

PX4 підтримує широкий спектр апаратних платформ: контролери польоту від Holybro, CUAV, mRo, Pixhawk, а також інші системи, що відповідають стандарту PX4. Окрім того, автопілот сумісний з програмним забезпеченням для симуляцій (наприклад, Gazebo або jMAVSim), що дозволяє розробникам тестувати алгоритми без ризику пошкодження апарата.

Система PX4 має тісну інтеграцію з наземною станцією QGroundControl — графічним інтерфейсом, за допомогою якого оператор може планувати польоти, контролювати стан апарата в реальному часі, переглядати телеметрію та налаштовувати параметри системи.

Ще однією ключовою перевагою PX4 є підтримка сучасних протоколів зв’язку, таких як MAVLink, що забезпечує гнучку комунікацію між БПЛА, наземною станцією та іншими системами. Це дозволяє легко розширювати

функціональність за рахунок додавання нових модулів, включаючи алгоритми штучного інтелекту, комп'ютерного зору, SLAM-навігації та інші інноваційні компоненти.

У практиці PX4 знаходить широке застосування в аграрному моніторингу, доставці вантажів, інспекції інфраструктури, пошуково-рятувальних операціях, а також у військових дослідженнях. Його відкритість і документованість робить цю платформу однією з найпривабливіших для досліджень і розробки нових рішень у сфері автономної робототехніки.

Таким чином, PX4 Autopilot є надійною, масштабованою та активно підтримуваною платформою, яка дозволяє створювати як прості, так і високотехнологічні безпілотні системи. Її використання значно спрощує процес проєктування, тестування та реалізації складних задач автономного управління літальними апаратами [10].

Інтерфейс програмного застосунку «PX4 Autopilot» зображено на рисунку 1.2.



Рис. 1.2 Інтерфейс програмного застосунку «PX4 Autopilot»

Mission Planner — це потужне програмне забезпечення для планування, моніторингу та аналізу польотів безпілотних літальних апаратів (БПЛА), створене на основі відкритого коду та тісно інтегроване з автопілотом ArduPilot. Це повноцінна наземна станція керування (GCS, Ground Control Station), яка

дозволяє як професіоналам, так і ентузіастам легко програмувати маршрути, налаштовувати параметри польоту, проводити калібрування систем, оновлювати прошивку, переглядати телеметрію та аналізувати результати польоту.

Mission Planner має зручний графічний інтерфейс, через який користувач може задавати точки маршруту на мапі, визначати поведінку дрона в кожній із них (наприклад, зупинитися, зробити знімок або повернутися на старт), а також програмувати автоматичні місії без потреби в складному кодуванні. Завдяки цьому програмне забезпечення часто використовується в сільському господарстві, геодезії, рятувальних операціях, технічному нагляді та наукових дослідженнях.

Крім планування польотів, Mission Planner виконує важливі функції з налаштування та діагностики. За допомогою інструментів калібрування можна точно налаштувати гіроскоп, акселерометр, компас, систему живлення та інші модулі БПЛА. Також програма дозволяє легко підключатися до автопілота через USB, Bluetooth, Wi-Fi або телеметрійні модулі, що забезпечує гнучкість у реальних польотних умовах.

Mission Planner підтримує двосторонню телеметрію, завдяки чому оператор може в режимі реального часу відстежувати місцезнаходження БПЛА, висоту, швидкість, стан батареї, режим польоту та інші параметри. Програма також виводить карту польоту та графіки, що дозволяє оперативно реагувати на зміни або в разі потреби втручатися в управління вручну.

Ще однією важливою функцією Mission Planner є можливість аналізу польотних логів. Після завершення місії користувач може завантажити детальні журнали польоту та переглянути, як працювали системи управління, чи виникали помилки, наскільки ефективно дрон дотримувався заданого маршруту. Ці можливості особливо корисні для розробників, тестувальників і дослідників, які займаються оптимізацією або виявленням збоїв у роботі БПЛА.

Mission Planner також має вбудовані інструменти для оновлення прошивки автопілота, вибору типу платформи (літак, коптер, глайдер тощо), активації додаткових функцій, таких як слідування за об’єктом (Follow Me), геозони, обмеження висоти та інші безпекові функції. Завдяки відкритій архітектурі й

активній спільноті, програму постійно оновлюють і доповнюють новими можливостями.

У підсумку, Mission Planner є універсальним інструментом для повноцінного управління БПЛА з автопілотом ArduPilot. Його функціональність охоплює весь життєвий цикл польоту — від планування місії до аналізу результатів — що робить його незамінним засобом як для початківців, так і для досвідчених операторів та розробників безпілотних систем [11].

Інтерфейс програмного застосунку «Mission Planner» зображено на рисунку 1.3.

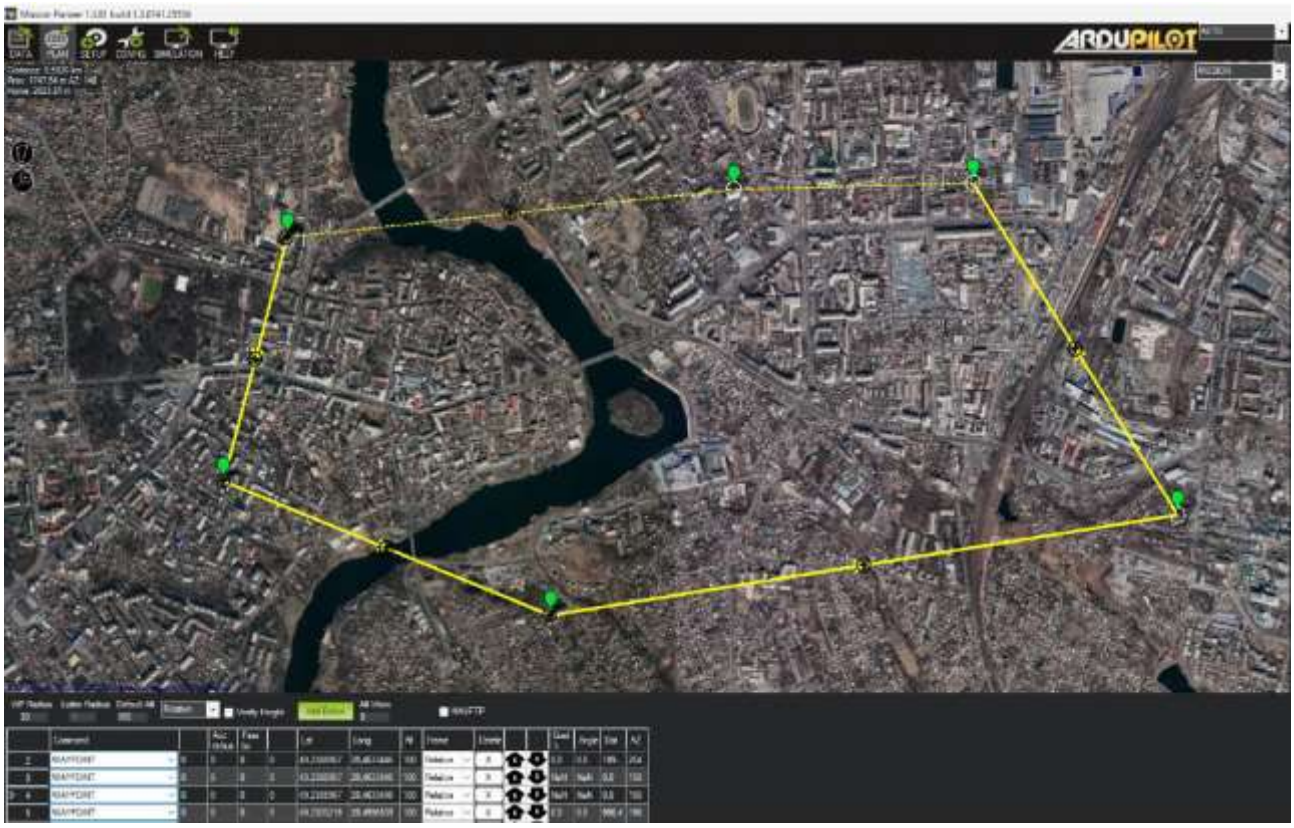


Рис. 1.3 Інтерфейс програмного застосунку «Mission Planner»

QGroundControl — це кросплатформна наземна станція управління (GCS — Ground Control Station), що забезпечує повноцінну взаємодію з безпілотними літальними апаратами в режимі реального часу. Вона створена з акцентом на простоту використання, універсальність та сумісність із популярними автопілотами, зокрема **PX4** і **ArduPilot**. З огляду на свою відкриту архітектуру, активну розробку та підтримку, QGroundControl є одним із найпопулярніших інструментів для роботи з дронами у професійному та дослідницькому середовищі.

Головною метою QGroundControl є забезпечення зручного інтерфейсу для планування польотів, налаштування параметрів апарата, моніторингу телеметрії та управління місіями в автоматичному або напівавтоматичному режимі. Інтерфейс програми інтуїтивно зрозумілий, сучасний і адаптований до різних пристроїв: він працює як на настільних ОС (Windows, macOS, Linux), так і на мобільних (Android, iOS).

Планування польотів у QGroundControl виконується на інтерактивній карті. Користувач може вказувати точки маршруту, типи завдань у кожній точці (наприклад, знімок, сканування території, пауза), а також задавати висоти, швидкість і напрямок польоту. Система автоматично перевіряє маршрут на предмет безпеки та обмежень (наприклад, мінімальна висота, дистанція до злітної точки). Крім того, QGroundControl дозволяє створювати складні місії з різними режимами — наприклад, автозліт, автопосадка, сканування ділянок за шаблоном тощо.

Програма підтримує роботу з багатьма типами платформ — мультикоптерами, літаками, VTOL-дронами, підводними апаратами та навіть автономними машинами на колесах. У кожному випадку вона надає відповідний набір інструментів та параметрів.

Важливою перевагою QGroundControl є можливість повного налаштування системи БПЛА без необхідності використання стороннього ПЗ. Наприклад, через інтерфейс можна калібрувати сенсори (компас, гіроскоп, акселерометр), налаштовувати параметри польотного контролера, керувати каналами RC-пульта, задавати межі польоту та параметри безпеки (наприклад, Return To Launch при втраті зв’язку).

QGroundControl підтримує телеметрійний зв’язок у реальному часі через протокол **MAVLink**, що забезпечує передачу великого обсягу даних з БПЛА на наземну станцію. Це дозволяє оператору бачити ключові параметри польоту: координати, швидкість, висоту, рівень заряду батареї, стан двигунів, температуру систем, режим польоту тощо. У разі потреби користувач може миттєво втрутитися у хід місії, змінити маршрут або перевести БПЛА в ручний режим.

Програма також має вбудовані функції запису та аналізу польотних логів. Після завершення місії можна переглянути хронологію дій, діагностувати проблеми або оптимізувати маршрут. Це особливо корисно для розробників, дослідників і операторів, які працюють у складних умовах або з нестандартними конфігураціями апаратів.

Однією з ключових переваг QGroundControl є її активна підтримка з боку розробників PX4. Завдяки цьому нові можливості автопілота одразу інтегруються в наземну станцію, а також підтримується сумісність з новими моделями контролерів та сенсорів.

Таким чином, **QGroundControl** є потужним і гнучким інструментом для повноцінного управління безпілотними апаратами. Завдяки відкритому коду, постійним оновленням, широкій сумісності та зручному інтерфейсу, ця платформа підходить як для новачків, так і для досвідчених інженерів, що працюють у галузі автономних систем [12].

Інтерфейс програмного застосунку «QGroundControl» зображено на рисунку 1.4.

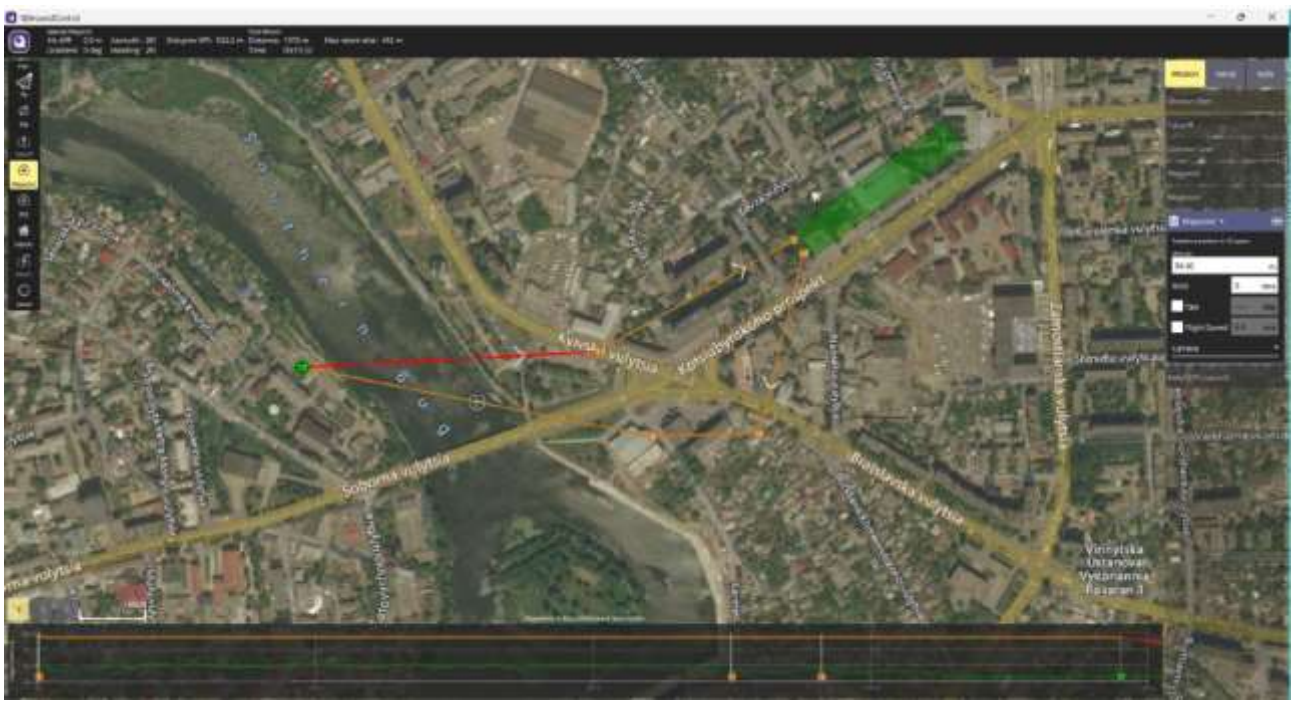


Рис.1.4 Інтерфейс програмного застосунку «QGroundControl»

DJI GS Pro (Ground Station Pro) — це потужний інструмент для професійного планування автоматичних польотів безпілотних літальних апаратів DJI, орієнтований насамперед на користувачів з галузей геодезії,

картографії, агромоніторингу, інспекції інфраструктури та пошуково-рятувальних робіт. Програма розроблена для планшетів на базі iPad і надає зручний графічний інтерфейс для точного налаштування місій, управління польотом у реальному часі та збору аерофотоданих із мінімальним втручанням оператора.

GS Pro забезпечує простий, інтуїтивно зрозумілий процес створення польотних місій. Користувач може прямо на мапі задавати маршрут або визначати полігони для автоматичного покриття території. Програма автоматично розраховує траєкторію польоту, з урахуванням заданих параметрів: висоти, перекриття знімків, швидкості дрону, кута камери, а також інших факторів, необхідних для якісного збору даних. Завдяки цьому GS Pro особливо популярний у сфері фотограмметрії та 3D-моделювання, де вимагається висока точність.

Однією з ключових функцій DJI GS Pro є "Photogrammetry Mode", який дозволяє створювати місії для аерозйомки з перекриттям кадрів, необхідним для побудови ортофотопланів, цифрових моделей місцевості (ЦММ) або 3D-реконструкції. Програма автоматично запускає камеру в заданих точках маршруту, синхронізуючи її з положенням дрона, що забезпечує точну геоприв’язку отриманих знімків.

GS Pro також підтримує режим "Waypoint Mission", у якому користувач самостійно визначає точки маршруту (waypoints), задає індивідуальні дії в кожній з них (наприклад, пауза, зміна висоти, поворот камери, знімок), і формує сценарії складних польотів. Такий режим корисний для інспекції об’єктів, зйомки фільмів, обльоту інфраструктури (лінії електропередач, мостів тощо), де необхідна гнучкість у побудові траєкторії.

Програма зберігає всі місії в хмарі або на пристрої, що дає змогу повторно запускати ті самі сценарії з однаковими параметрами — важливо, коли потрібно регулярно моніторити одну й ту саму територію.

GS Pro забезпечує зв'язок із дроном через стандартне обладнання DJI, включаючи популярні моделі, як-от Phantom 4 Pro, Inspire 2, Matrice 200/300, Mavic 2 Enterprise, а також сумісні камери Zenmuse. Підключення здійснюється

через контролер з підключеним iPad, що дає змогу керувати дроном, бачити відеопотік у реальному часі та контролювати стан акумулятора, GPS, компаса, сенсорів уникнення перешкод тощо.

З точки зору безпеки, GS Pro дозволяє налаштовувати межі польоту (геозони), висоту повернення додому (RTL), контроль зв'язку та обробку позаштатних ситуацій (наприклад, втрату сигналу або розряд батареї). Система також передбачає ведення журналів польотів, які можна використовувати для подальшого аналізу.

Програмне забезпечення інтегрується з деякими зовнішніми платформами обробки даних, наприклад, Pix4D, DroneDeploy, або DJI Terra, що дозволяє організувати повний цикл від планування місії до отримання готового геопросторового продукту.

У підсумку, DJI GS Pro є надійним, точним та професійним інструментом для користувачів екосистеми DJI, яким потрібна гнучкість, автоматизація та висока якість збору аероданих. Його функціональність значно розширює можливості типових польотів, роблячи дрони DJI ефективними інструментами у складних прикладних сценаріях [13].

Інтерфейс програмного застосунку «DJI GS Pro» зображено на рисунку 1.5.

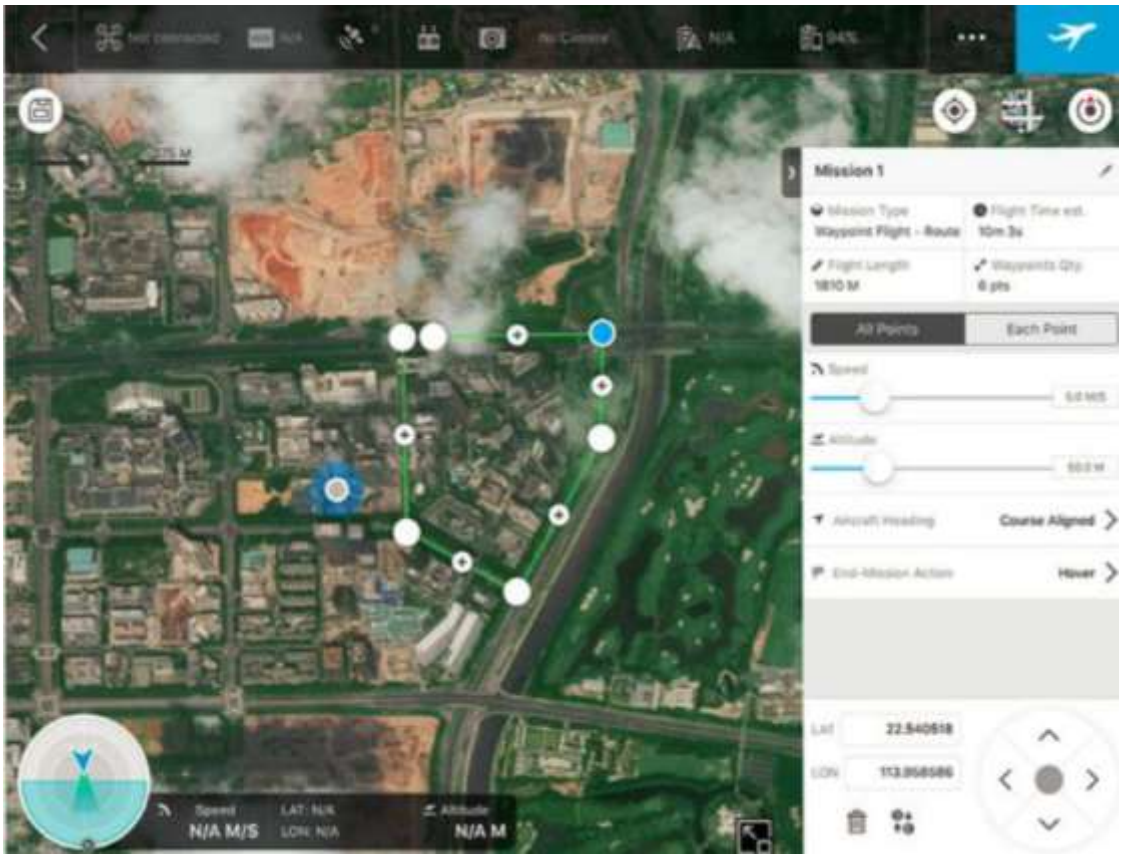


Рис. 1.5 Інтерфейс програмного застосунку «DJI GS Pro»

Розглянувши ці аналоги було створено таблицю 1.1 для порівняння їхнього функціоналу з програмним забезпеченням «DronePathfinder».

Аналізуючи вищенаведені дані, «DronePathfinder» має вищий сумарний коефіцієнт відносно аналогів: «PathPlanner» на 80%, «PX4 Autopilot» на 40%, «Mission Planner» на 20%, «QgroundControl» на 20%, «DJI GS Pro» на 20%.

1.1. Функціональні характеристики програмних застосунків

Характеристика	**PathPlanner**	**PX4 Autopilot**	**Mission Planner**	**QGroundControl**	**DJI GS Pro**
Тип	Планувальник траєкторій	Автопілот/прошивка	Наземна станція управління (GCS)	Наземна станція управління (GCS)	Мобільна станція управління
Призначення	Генерація та симуляція оптимальних траєкторій	Розрахунок, стабілізація та контроль польоту	Планування, моніторинг і налаштування БПЛА	Універсальне управління місіями	Планування місій для дронів DJI
Підтримка апаратів	Симулятори,	PX4-сумісні БПЛА	ArduPilot-based (наприклад,	PX4, ArduPilot, мультиплатформено	Тільки дрони DJI (Phantom,

Характеристика	PathPlanner	PX4 Autopilot	Mission Planner	QGroundControl	DJI GS Pro
	платформи із ROS		Copter, Plane)		Mavic, Matrice тощо)
Тип доступу / код	Відкритий (Open Source)	Відкритий (Open Source)	Відкритий (Open Source)	Відкритий (Open Source)	Закритий, комерційний (DJI)
Платформи (ОС)	Windows, Linux	Вбудовується у БПЛА (на контролерах)	Windows	Windows, macOS, Linux, iOS, Android	iOS (iPad) тільки
Автоматизація місій	✔ Генерація автономних шляхів	✔ Контроль виконання	✔ Автоматизовані маршрути	✔ Потужні місії з логікою	✔ Автоматичні польоти по зонам
Інтерфейс користувача	Технічний, інженерний	CLI / параметри автопілота	Графічний, менш сучасний	Інтуїтивний, сучасний	Дуже зручний, простий для iPad
Планування 2D/3D місій	✔ З 3D-візуалізацією	✖ (Низький рівень доступу)	✔ Переважно 2D	✔ 2D/3D обліт, точкове планування	✔ Зона обльоту, об’ємне сканування
Реальний контроль під час польоту	✖ (не призначений для живого управління)	✔ (через зовнішнє GCS)	✔ Повний контроль	✔ Повний контроль	✔ Обмежений — більше для автоматичних місій
Підтримка телеметрії / логів	✖ (немає прямого зв’язку з БПЛА)	✔ (через MAVLink)	✔ Детальні логи, аналіз	✔ Графіки, аналіз, телеметрія	✔ Мінімальний перегляд після польоту
Гнучкість налаштувань системи	✔ Висока (параметри траєкторій, алгоритми)	✔ Дуже висока (низькорівневий контроль)	✔ Повний доступ до параметрів автопілота	✔ Повна підтримка налаштувань сенсорів, каналів	✖ Мінімальна (переважно для кінцевих користувачів)
Основні користувачі	Дослідники, інженери, розробники	Розробники автопілотів, інтегратори	Хобісти, технічні спеціалісти	Універсально: дослідники, оператори, комерція	Комерційні користувачі, інспекція, агро, фото
Сфери застосування	Навігація, моделювання, автономні роботи	Будь-який тип безпілотників	Сільське господарство, хобі, технічна інспекція	Геодезія, промисловість, навчання, дослідження	Картографування, агроаналіз, фото- і відеозйомка

На сучасному етапі розвитку безпілотних технологій існує широкий спектр програмного забезпечення, що забезпечує автоматизацію планування, оптимізацію маршруту та управління польотом безпілотних літальних апаратів (БПЛА). Кожна з розглянутих систем — PathPlanner, PX4 Autopilot, Mission Planner, QGroundControl та DJI GS Pro — виконує свою специфічну роль у комплексі навігації та управління БПЛА, при цьому має власні переваги та обмеження.

PathPlanner є інженерним інструментом, призначеним для побудови оптимальних траєкторій руху, головним чином у симуляційних середовищах або в рамках автономних роботизованих систем. Його основною перевагою є високий рівень контролю над алгоритмами побудови маршруту, що робить його цінним для досліджень та розробки інтелектуальних навігаційних систем.

PX4 Autopilot виступає як високонадійне програмне ядро для керування польотом. Його відкрита архітектура дозволяє легко адаптувати систему до різних типів апаратів — від мультикоптерів до літаків і VTOL-платформ. Завдяки гнучкості, підтримці сучасних стандартів (MAVLink) і активній спільноті розробників, PX4 є одним із провідних рішень для побудови автономних БПЛА.

Mission Planner — це класична наземна станція для роботи з автопілотом ArduPilot, яка дозволяє планувати польоти, налаштовувати апарат і відслідковувати телеметрію. Незважаючи на дещо застарілий інтерфейс, програма пропонує велику кількість функцій, зокрема розширене налаштування параметрів і роботу з логами, що робить її корисною для технічно підготовлених користувачів.

QGroundControl є найбільш універсальним середовищем управління, яке підтримує як PX4, так і ArduPilot. Завдяки сучасному інтерфейсу, кросплатформеності, широким можливостям налаштування та інтуїтивному плануванню місій, ця система є ефективним інструментом як для новачків, так і для досвідчених операторів БПЛА. Її гнучкість робить її придатною для широкого спектру завдань — від дослідницьких до промислових.

DJI GS Pro, своєю чергою, орієнтована на кінцевих користувачів дронів DJI. Це комерційний продукт, який забезпечує просте та швидке створення автоматизованих місій, головним чином для аерофотозйомки, 3D-картографії та агромоніторингу. Система має високий рівень зручності, але обмежена у гнучкості та розширенні через закриту екосистему DJI.

Таким чином, вибір тієї чи іншої системи залежить від завдань, рівня підготовки користувача та типу використовуваного БПЛА. Для дослідницьких та експериментальних цілей найбільш доцільним виглядає комбінація PX4 + QGroundControl або ArduPilot + Mission Planner. Для комерційного використання з дронами DJI найкращим рішенням є DJI GS Pro, що забезпечує швидке впровадження місій без глибокого технічного налаштування. PathPlanner може ефективно застосовуватись як допоміжний модуль у системах, що вимагають автономної навігації з врахуванням складних обмежень середовища.

1.3 Вплив алгоритмів знаходження оптимального шляху на ефективність безпілотного літального апарату

Алгоритми знаходження оптимального шляху відіграють критично важливу роль в ефективному функціонуванні безпілотних літальних апаратів (БПЛА), особливо у контексті автономного польоту. Від обраного алгоритму залежить не лише маршрут руху дрона, а й низка ключових показників — енергоефективність, тривалість місії, безпечність навігації, якість виконання завдань та здатність працювати у складних або динамічних умовах.

1. Оптимізація маршруту — зменшення витрат енергії. Один із найбільш очевидних впливів алгоритмів планування шляху — це мінімізація витрат енергії. БПЛА має обмежений ресурс акумулятора, тому побудова найкоротшого або найменш затратного за енергією маршруту дозволяє збільшити час у повітрі, зменшити ризик аварійного повернення та підвищити загальну надійність місії. Алгоритми, що враховують рельєф, вітер, заборонені зони та інші фактори (наприклад, A*, D*, Theta*, RRT*, PRM), дозволяють ефективно планувати маршрут з урахуванням реальних умов середовища.

2. Безпечність польоту. Алгоритми прокладання шляху значно впливають на безпечність виконання завдань. У разі наявності перешкод (будівлі, дерева, повітряні зони обмеження польотів тощо), важливо забезпечити не просто короткий маршрут, а такий, що гарантує уникнення небезпеки. Планування з урахуванням тривимірних перешкод або зміни обстановки (наприклад, для пошуково-рятувальних дронів) забезпечує більшу автономність та захищеність БПЛА.

3. Адаптивність до змін середовища. У реальних умовах БПЛА часто стикається з динамічними змінами: з’являються нові перешкоди, змінюються погодні умови, змінюється місце розташування цілей. Алгоритми з можливістю реального перепланування маршруту (replanning) у польоті дозволяють дрону гнучко реагувати на події без втручання оператора. Це особливо важливо для військових, рятувальних чи дослідницьких місій.

4. Якість виконання завдань. Ефективне планування маршруту також впливає на точність та повноту виконання завдань. Наприклад, при аерофотозйомці оптимальний маршрут із заданим перекриттям кадрів забезпечує якісну фотограмметрію, а при інспекції об’єктів — повний огляд без пропусків. Неправильно спланований маршрут може призвести до втрати даних або необхідності повторного виконання місії.

5. Зменшення часу місії. Швидкість виконання завдань прямо залежить від часу, необхідного для обльоту всіх контрольних точок або зони інтересу. Оптимізація маршруту дозволяє значно зекономити час, що критично в умовах обмеженого вікна роботи (наприклад, через погодні умови, тривалість світлового дня або загрозу ворожої активності).

6. Можливість масштабування та автономності. Завдяки використанню алгоритмів планування шляху, БПЛА може виконувати складні місії без постійного контролю оператора, що дозволяє розгортати рої дронів або здійснювати масштабовані операції (наприклад, у сільському господарстві чи моніторингу великих територій). Це відкриває шлях до повної автономізації повітряних роботизованих систем.

Таким чином, алгоритми знаходження оптимального шляху мають фундаментальний вплив на ефективність, безпечність та автономність роботи БПЛА. Вони визначають, наскільки економно буде витрачатися ресурс, як швидко та точно буде виконано завдання, чи здатен дрон діяти самостійно у складних умовах. Тому вибір і реалізація відповідного алгоритму повинні відповідати характеру місії, навколишньому середовищу та рівню автономності системи керування. У поєднанні з сучасними навігаційними системами та сенсорами, правильно реалізоване планування маршруту значно підвищує загальну ефективність використання безпілотних літальних апаратів у найрізноманітніших галузях.

1.4 Постановка задач дослідження

У сучасних умовах інтенсивного розвитку безпілотних літальних апаратів (БПЛА) дедалі більшої актуальності набуває питання ефективної та безпечної навігації у просторі. Зростає потреба у забезпеченні повної або часткової автономності польоту, зокрема при виконанні завдань у складних умовах: міській забудові, лісовій місцевості, зонах обмеженого доступу або динамічному середовищі. Одним із ключових елементів автономного управління є здатність БПЛА обчислювати та реалізовувати оптимальний маршрут руху з урахуванням обмежень, цілей та змінних зовнішніх факторів.

Задача знаходження оптимального шляху полягає у визначенні такої траєкторії, яка дозволить БПЛА дістатися до цільової точки з мінімальними витратами ресурсу (енергії, часу, ризику), при цьому уникнувши перешкод та дотримуючись заданих обмежень (геометричних, динамічних, нормативних тощо).

З огляду на викладене, дослідження має на меті розробку, аналіз або удосконалення алгоритмів, які дозволяють забезпечити оптимальне планування маршруту польоту БПЛА. При цьому слід врахувати особливості реального середовища, тип апарату, наявність сенсорів, рівень автономності системи управління та специфіку поставлених задач.

Проаналізувавши переваги та недоліки існуючих системи знаходження шляху безпілотного літального апарату було визначено наступні завдання, які необхідно виконати для розробки мобільної застосунку:

− розробити загальний алгоритм системи;

− розробити метод знаходження оптимального шляху безпілотного літального апарату на основі алгоритмів Дейкстри та A*;

− розробити метод врахування характеристик безпілотника;

− розробити зручний та адаптивний графічний інтерфейс мобільного застосунку;

− розробити модуль для локального збереження маршрутів на пристрої користувача;

− здійснити тестування мобільного застосунку згідно поставлених задач;

РОЗДІЛ 2 РОЗРОБКА МЕТОДІВ, МОДЕЛІ ТА АЛГОРИТМІВ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Аналіз вхідних та вихідних даних системи

Проєктування мобільного застосунку, що виконує функцію побудови оптимального маршруту для безпілотного літального апарату, потребує глибокого розуміння того, з якими даними система працює. Цей аналіз охоплює дві ключові категорії: вхідні дані, які необхідні для запуску обчислювального процесу маршруту, та вихідні дані, які є результатом роботи застосунку й передаються користувачу або бортовій системі управління дроном.

Усі вхідні дані, які використовує система, повинні бути релевантними, точними та мати зрозумілу структуру. Насамперед, мобільний застосунок отримує географічну інформацію — координати точки старту та пункту призначення польоту. Це є базовою передумовою для побудови будь-якого маршруту. Як правило, ці дані надаються у форматі глобальних координат GPS (широта, довгота, висота).

Додатково можуть вводитися проміжні точки — waypoints, які мають бути відвідані дроном під час виконання місії. Такими точками можуть бути об’єкти зйомки, зони збору даних, або контрольні точки моніторингу.

Одним із важливих типів вхідної інформації є обмеження та умови польоту. До них належать заборонені зони, зони з обмеженою висотою, маршрути, які необхідно оминати через наявність перешкод або юридичні обмеження. Наприклад, у містах такі зони можуть охоплювати території поблизу аеропортів, урядових будівель, шкіл тощо. У додатку ці дані можуть автоматично підвантажуватись із відкритих джерел або вводитись вручну користувачем.

Важливими є також параметри самого безпілотного літального апарату, який використовуватиметься. До них належать технічні характеристики: максимальна швидкість, допустима висота польоту, максимальна тривалість роботи на одному заряді, маса корисного навантаження, а також модель автопілоту. Ці дані дозволяють алгоритму розраховувати реалістичні маршрути, які не перевищуватимуть фізичних можливостей пристрою.

Крім того, система має враховувати зовнішні умови польоту. Одним із критичних чинників є метеорологічна ситуація — напрямок і швидкість вітру, температура повітря, наявність опадів. Вітер, наприклад, може істотно вплинути на споживання енергії: зустрічний вітер потребує додаткових зусиль і скорочує запас ходу, а попутний — навпаки. Такі дані можуть інтегруватися в застосунок через погодні API або вводитись користувачем вручну.

Не менш важливо визначити тип завдання, яке виконує дрон. Якщо це доставка вантажу, то пріоритетом може бути мінімальний час. Якщо — зйомка або моніторинг, то потрібно забезпечити перекриття зон спостереження або проходження через задані ділянки з певною висотою та швидкістю. Таким чином, система повинна адаптувати маршрут не лише до середовища, а й до змісту місії.

Після того як застосунок обробив усю вхідну інформацію, він повинен сформувати набір результатів, що можуть бути використані для запуску польоту або перегляду маршруту користувачем. Найголовнішим результатом є побудований маршрут. Він формується у вигляді послідовності координат з прив’язкою до висоти — так званих точок маршруту або waypoints. У більш складних сценаріях маршрут може включати криволінійні траєкторії, з урахуванням динаміки руху дрона.

До маршруту додається інформація про кожен етап польоту: орієнтовний час польоту до кожної точки, залишок заряду батареї на кожному етапі, швидкість переміщення, ухил, висота над рівнем землі. Якщо польотна місія включає виконання дій (наприклад, фотофіксацію або скидання вантажу), ці команди також повинні бути прописані у маршруті.

Інтерфейс застосунку, як правило, має забезпечувати візуалізацію побудованого маршруту на карті з позначенням важливих точок та перешкод. Це дозволяє користувачу перевірити маршрут до запуску, а також внести зміни, якщо того вимагає ситуація.

Ще один важливий тип вихідних даних — це експорт місії у форматі, придатному для конкретного автопілота. Для апаратів з PX4 або ArduPilot, наприклад, використовуються маршрути у форматі MAVLink. У випадку дронів

DJI застосовуються інші формати. Застосунок повинен мати можливість зберігати місії для подальшого завантаження на дрон або передавати їх по бездротовому зв’язку.

Крім цього, користувач повинен отримати повідомлення про ризики, якщо такі виявлені під час побудови маршруту — наприклад, перетин забороненої зони, недостатній заряд акумулятора для повного проходження маршруту, перевищення максимально допустимої висоти або швидкості.

Аналіз вхідних і вихідних даних є критично важливим етапом при розробці мобільного застосунку для побудови оптимального маршруту БПЛА. Саме на підставі коректно зібраної вхідної інформації система здатна генерувати точні, безпечні та енергоефективні маршрути, що враховують особливості апарату, середовища та задачі місії. Якість вихідних даних, своєю чергою, визначає, наскільки зручно користувачеві буде взаємодіяти із системою, а також наскільки точно автопілот зможе виконати політ згідно з запланованим сценарієм. Тому формалізація, валідація та обробка даних на вході й виході є основою успішної реалізації подібного застосунку.

2.2 Розробка загального алгоритму роботи системи та методу визначення оптимального шляху на основі алгоритмів Дейкстри та A*

Для розробки мобільної застосунку для знаходження оптимального шляху необхідно розробити загальний алгоритм системи, згідно якого буде працювати мобільний застосунок (рисунки 2.1 – 2.2).

Після відкриття мобільного застосунку користувачу доступно 8 основних дій.

Робота мобільного застосунку «DronePathfinder» розпочинається зі стандартного етапу ініціалізації. Після запуску додатку здійснюється його завантаження, під час якого завантажуються основні компоненти інтерфейсу та функціональні модулі.

Після завершення початкового завантаження, система очікує дії користувача. Ключовим етапом є вибір розділу «Маршрути». Саме цей розділ

відповідає за побудову та перегляд маршрутів для безпілотного літального апарату.

- Якщо користувач не обирає розділ «Маршрути», система перенаправляє його до іншої частини додатку, передбаченої для альтернативного функціоналу (наприклад, налаштування, перегляд статистики польотів тощо). Це відображено через вихід до іншої гілки алгоритму (перехід на сторінку 5 або 17, згідно з вказівками на схемі).
- Якщо ж користувач обирає розділ «Маршрути», додаток ініціює завантаження відповідної розмітки інтерфейсу, яка дає змогу розпочати процес створення маршруту, редагування наявних або запуску оптимізаційних алгоритмів. Після цього виконується перехід до наступного етапу логіки застосунку (на сторінку 6, відповідно до логіки блок-схеми).

Таким чином, загальний алгоритм реалізує логіку умовного розгалуження дій користувача на основі його вибору в головному інтерфейсі. Подальші дії залежать від контексту: якщо користувач зацікавлений у побудові маршруту — він потрапляє у відповідний функціональний блок, де вже може взаємодіяти з картою, задавати точки, обирати оптимізаційний метод тощо.

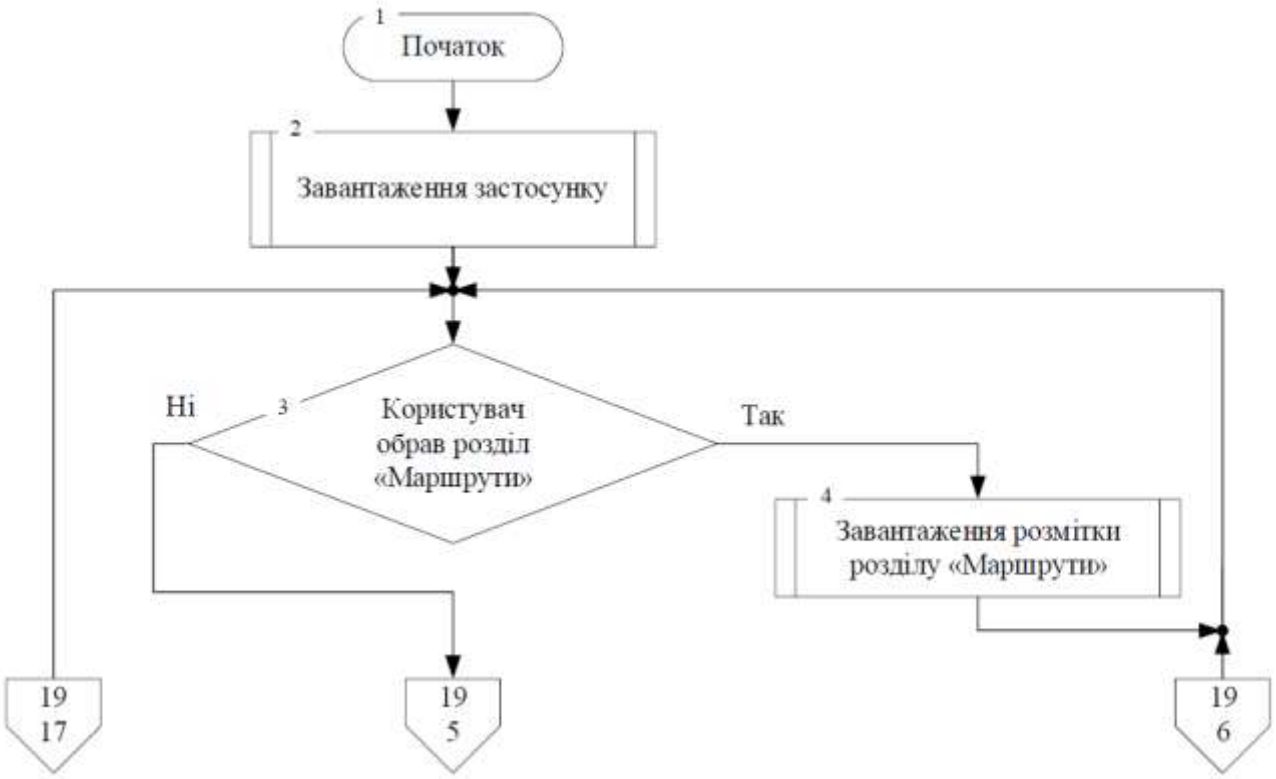


Рис. 2.1 Загальний алгоритм роботи застосунку «DronePathfinder»

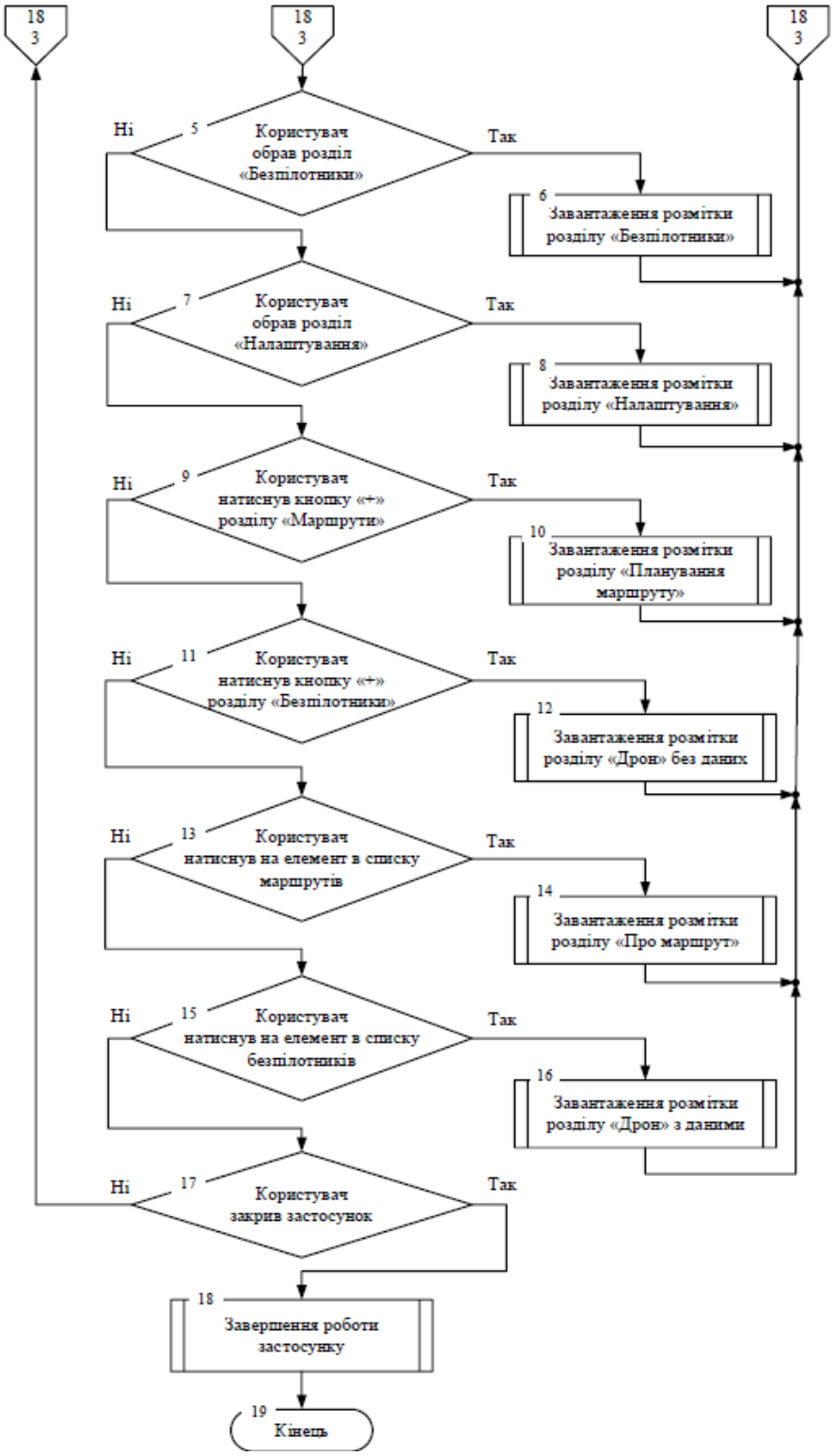


Рис. 2.2 Продовження загального алгоритму застосунку «DronePathfinder»

Головною задачею мобільної застосунку є визначення та побудова оптимального шляху для безпілотного літального апарату.

Після завантаження розділу «Маршрути» користувач має змогу взаємодіяти з іншими частинами інтерфейсу програми. У випадку, якщо розділ «Маршрути»

не був обраний одразу, додаток перевіряє послідовно інші можливі дії користувача:

1. Вибір розділу «Безпілотники» — якщо користувач обирає цей розділ, завантажується відповідна розмітка, що містить інформацію про наявні дрони, їх характеристики та доступні функції керування.
2. Вибір розділу «Налаштування» — у цьому випадку відкривається інтерфейс для конфігурації параметрів застосунку, підключення до дронів, встановлення одиниць виміру, мовних налаштувань тощо.
3. Натискання кнопки «+» у розділі «Маршрути» — активує процес планування нового маршруту. Завантажується відповідний інтерфейс для введення точок маршруту, параметрів висоти, швидкості та оптимізації.
4. Натискання кнопки «+» у розділі «Безпілотники» — відкривається форма додавання нового дрона до системи. У цьому випадку завантажується шаблон порожнього профілю дрона (розділ «Drone без даних»), куди можна внести необхідну інформацію.
5. Натискання на елемент списку маршрутів — відображає детальну інформацію про обраний маршрут. Завантажується розділ «Про маршрут», де користувач може переглянути дані польоту, характеристики шляху, а також редагувати маршрут або запускати його на виконання.
6. Натискання на елемент списку безпілотників — аналогічно, відкривається інформація про конкретний дрон. Завантажується розділ «Drone з даними», де можна переглядати технічні характеристики, статистику польотів та проводити діагностику.
7. Закриття застосунку — якщо користувач завершує роботу із застосунком, відбувається завершення усіх процесів, очищення тимчасових даних і вихід із програми.

Таким чином, алгоритм охоплює повний цикл дій користувача: від запуску застосунку та навігації по розділах до взаємодії з даними безпілотників і маршрутів, а також завершення роботи. Кожен крок супроводжується завантаженням відповідного інтерфейсу, що забезпечує зручність та послідовність користування.

Метод findShortestPath використовує алгоритм на вибір користувача – алгоритм Дейкстри або алгоритм A* – для знаходження найкоротшого шляху між точками на карті. Ці алгоритми є одними з найвідоміших алгоритмів пошуку шляху в графі, які знаходять шлях з найменшою сумарною вагою між двома вершинами.

Для ілюстрації роботи алгоритму було створено блок-схему, яка детально показує кожен крок процесу, від ініціалізації до відтворення шляху (рисунки 2.3 – 2.5).

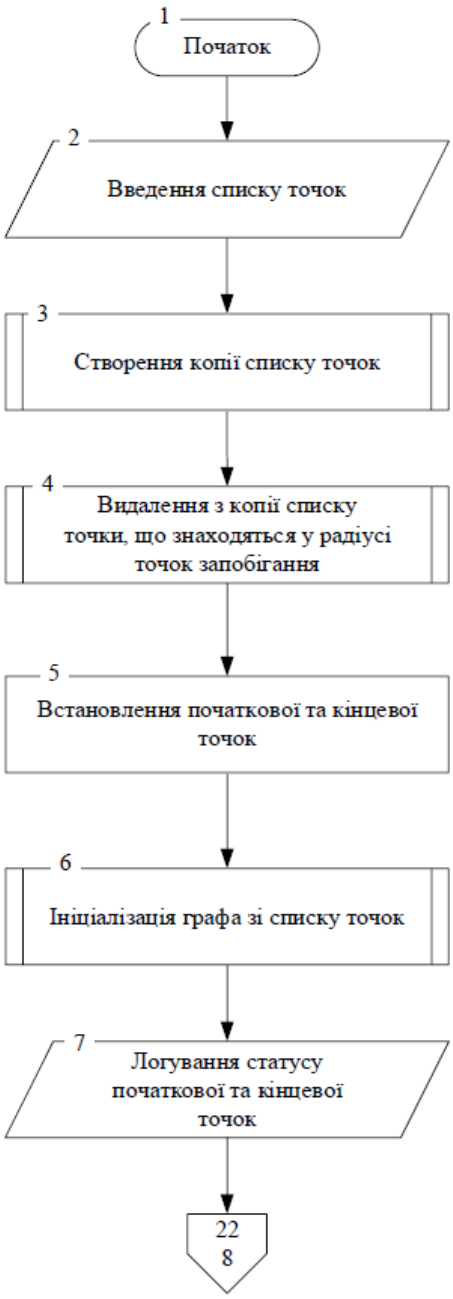


Рис. 2.3. Блок-схема методу визначення оптимального шляху на основі алгоритму Дейкстри (частина 1)

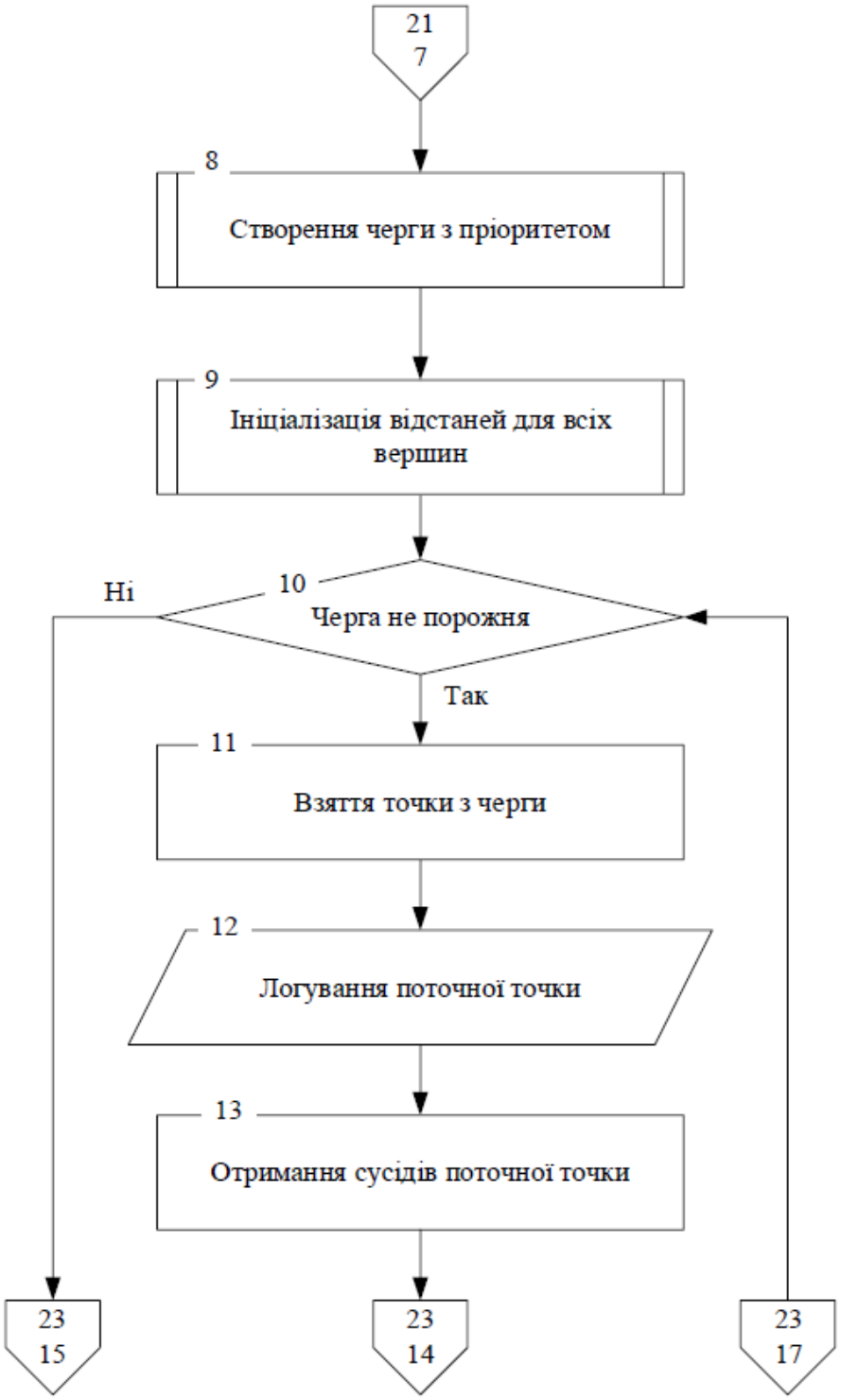


Рис. 2.4. Блок-схема методу визначення оптимального шляху на основі алгоритму Дейкстри (частина 2)

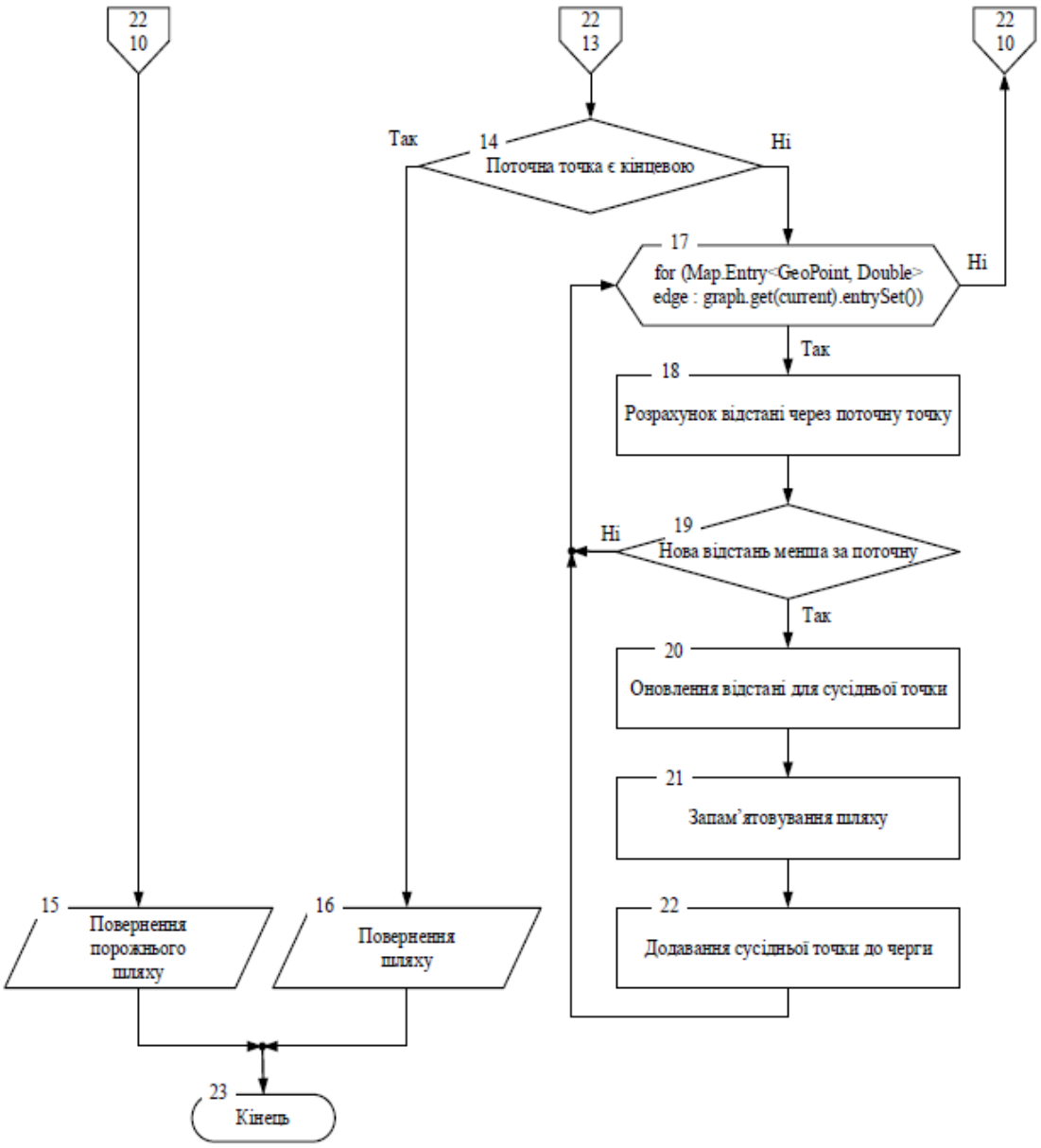


Рис.2.5 Блок-схема методу визначення оптимального шляху на основі алгоритму Дейкстри (частина 3)

Цей алгоритм дозволяє повністю автоматизувати процес визначення оптимального шляху. Його ефективність робить придатним для використання в реальному часі, навіть для великих графів, а універсальність і масштабованість роблять його незамінним для вирішення широкого спектру задач, пов'язаних з маршрутизацією.

2.3 Розробка методу врахування характеристик безпілотника та моделі системи

Іншою не менш важливою задачею мобільної застосунку є врахування характеристик безпілотника та надання рекомендацій.

Надання рекомендацій відбувається автоматично на основі статусу маршруту, що визначається при створенні маршруту і оновлюється при доданні безпілотників до маршруту. Кожен маршрут обов'язково має один із чотирьох статусів:

− NO_DRONE – вказує на відсутність дрона;

− ROUTE_TOO_LONG – вказує на те, що маршрут занадто довгий для дрона;

− OVERWEIGHT – означає, що дрон перевантажений;

− GOOD – вказує на те, що дрон готовий до польоту.

Спочатку відбувається перевірка на наявність дрону: якщо до маршруту не доданий дрон, то метод повертає маршруту статус NO_DRONE і надає користувачу повідомлення з рекомендацією: «Маршруту не призначений дрон! Призначте дрон натиснувши кнопку внизу екрану.».

В іншому випадку, коли до маршруту доданий дрон, то відбувається ще декілька перевірок:

1. Якщо максимальна дальність польоту дрона менша за обраховану довжину маршруту, метод повертає маршруту статус ROUTE_TOO_LONG і надає повідомлення з рекомендацією: «Довжина маршруту більша за дальність польоту дрону! Призначте інший дрон натиснувши кнопку внизу екрану.».

2. Якщо вантажопідйомність дрона менше за вказану вагу вантажу, метод повертає маршруту статус OVERWEIGHT і надає повідомлення з рекомендацією: «Вага вантажу більша за вантажопідйомність дрону! Призначте інший дрон натиснувши кнопку внизу екрану.».

3. Якщо жодна з попередніх умов не виконується, метод повертає маршруту статус GOOD.

На рисунку 2.6 зображено блок-схему методу врахування характеристик безпілотника.

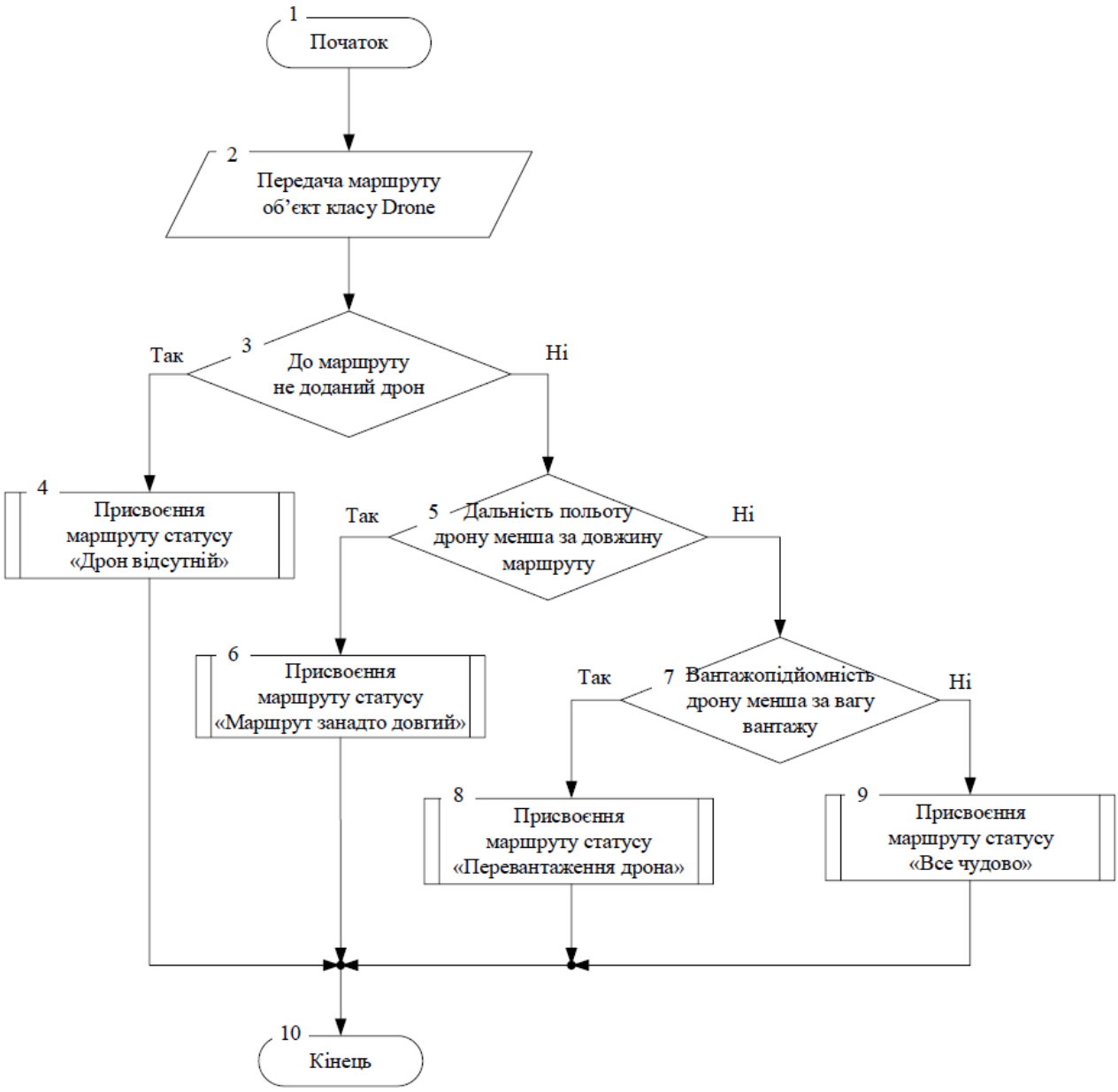


Рис. 2.6. Блок-схема методу врахування характеристик безпілотника

UML діаграма діяльності є ключовим інструментом для візуалізації та планування взаємодії користувача з мобільною системою. Вона допомагає розробникам зрозуміти послідовність дій та взаємозв'язки між різними модулями, а також сприяє ефективному проєктуванню користувацького інтерфейсу [14].

Для наочного представлення функціонування програмних модулів мобільної системи було розроблено UML діаграму діяльності (рисунок 2.7).

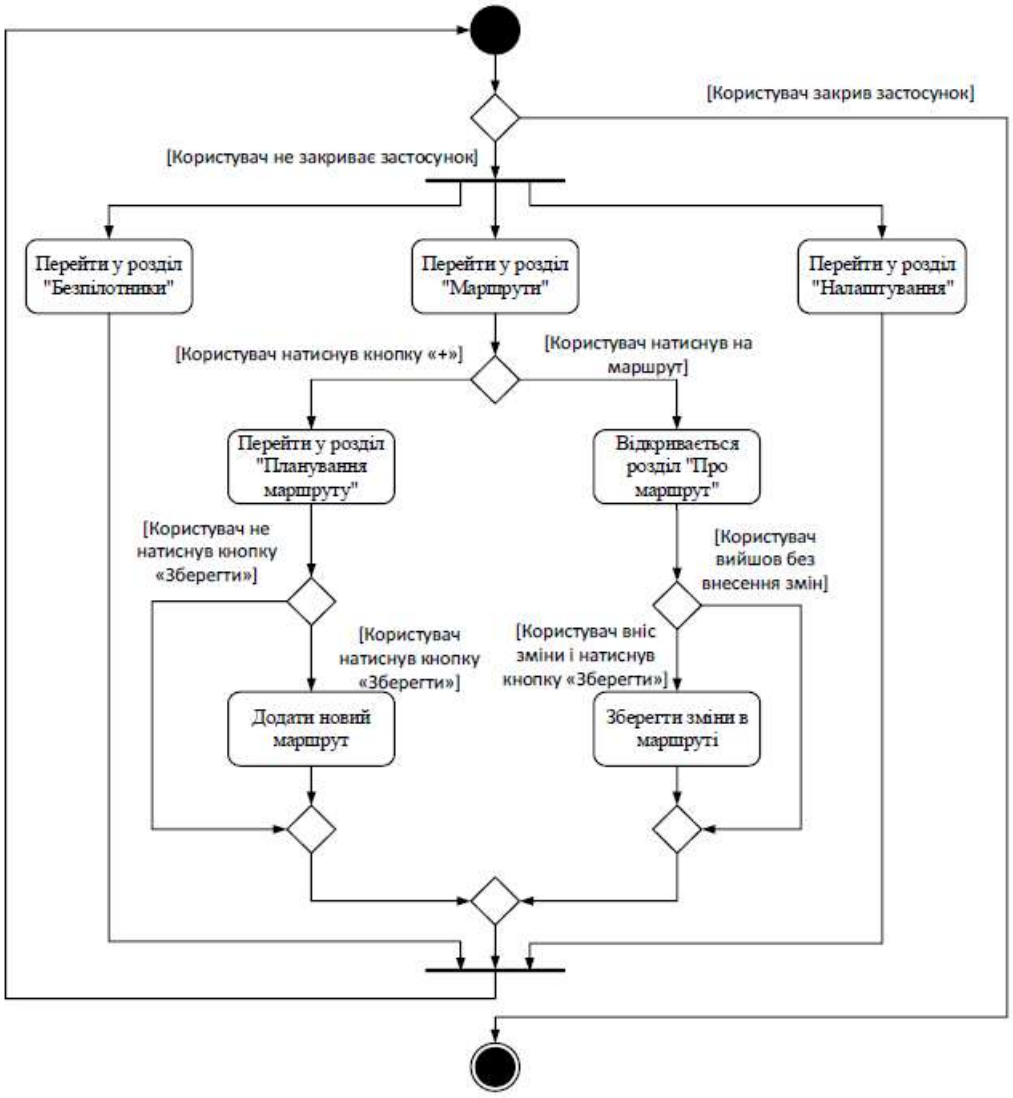


Рис. 2.7. Діаграма діяльності мобільного застосунку

2.4 Аналіз і обґрунтування вибору мови програмування та обґрунтування вибору засобу відображення мапи

У світі мобільної розробки, вибір мови програмування може мати значний вплив на процес створення застосунку. При розробці мобільних застосунків для платформи Android зазвичай використовуються дві мови, які часто порівнюються – це Java та Kotlin. Обидві мови мають свої переваги та недоліки, і вибір між ними залежить від багатьох факторів [15].

Java є об'єктно-орієнтованою мовою програмування загального призначення, яка широко використовується для створення програмного забезпечення різного рівня складності, зокрема мобільних застосунків для

операційної системи Android. Ця мова була офіційною мовою програмування Android з моменту його запуску у 2008 році, і хоча згодом з’явилися альтернативи, такі як Kotlin, Java досі залишається популярним інструментом серед розробників мобільних застосунків.

Однією з основних причин популярності Java у сфері Android-розробки є її повна інтеграція з Android SDK (Software Development Kit), який містить набір бібліотек, інструментів і документації, необхідних для створення Android-застосунків. Усі основні компоненти операційної системи Android, зокрема класи для роботи з інтерфейсом користувача, базами даних, мережею, мультимедіа та апаратними ресурсами, мають реалізацію на Java або сумісні з нею через відповідні API.

Java підтримує всі основні принципи об'єктно-орієнтованого програмування, такі як інкапсуляція, спадкування, поліморфізм і абстракція. Завдяки цьому розробники можуть структурувати застосунок у вигляді незалежних модулів, що значно спрощує процес його масштабування, супроводу та тестування. Крім того, Java має вбудовані механізми керування пам’яттю (через автоматичний збирач сміття), що сприяє стабільності роботи мобільного застосунку навіть на пристроях з обмеженими ресурсами.

Процес створення Android-застосунку на Java передбачає використання таких основних компонентів, як Activity, Fragment, Service, BroadcastReceiver та ContentProvider. Activity відповідає за відображення інтерфейсу користувача і взаємодію з ним. Fragment дозволяє розбивати інтерфейс на окремі логічні частини, що сприяє гнучкому управлінню екранним простором. Компонент Service використовується для виконання фонових задач, які не залежать від активного інтерфейсу користувача.

Для розробки Android-застосунків на Java зазвичай використовується інтегроване середовище розробки Android Studio, яке підтримується компанією Google. Воно надає зручні інструменти для написання коду, налагодження, тестування та оптимізації застосунку. Середовище також підтримує емулятори Android-пристроїв, що дає змогу перевірити роботу програми без фізичного пристрою. Система збирання Gradle, яка є частиною Android Studio, автоматизує

процес компіляції, керування залежностями та побудови пакету застосунку (APK-файлу).

Попри те, що Java вважається надійною та зрілою мовою, вона має певні обмеження. Одним із недоліків є так звана "вербозність" мови, тобто необхідність написання великої кількості коду навіть для реалізації простих дій. Це може знижувати продуктивність розробки порівняно з сучаснішими мовами, такими як Kotlin, яка є більш лаконічною та гнучкою. Проте завдяки великій спільноті розробників, широкій базі знань, наявності численних прикладів та бібліотек, Java залишається стабільною основою для побудови надійних Android-застосунків.

Узагальнюючи, можна зазначити, що Java є потужним інструментом для створення мобільних застосунків під Android завдяки своїй стабільності, повній сумісності з Android SDK, об'єктно-орієнтованій природі та підтримці з боку великої спільноти. Для проєктів, що орієнтуються на перевірені технології та стабільну архітектуру, Java залишається актуальним і доцільним вибором.

Kotlin є сучасною статично типізованою мовою програмування, яка була офіційно визнана компанією Google як основна мова для розробки Android-застосунків у 2017 році. Вона була створена компанією JetBrains з метою усунення недоліків мови Java, зберігаючи при цьому сумісність з існуючими Java-бібліотеками та інфраструктурою. Kotlin повністю сумісний з Android SDK та має розширену підтримку в Android Studio — офіційному середовищі розробки Android-застосунків.

Однією з основних переваг Kotlin є його лаконічний і виразний синтаксис, який дозволяє значно скоротити обсяг коду без втрати функціональності або читабельності. Це не лише підвищує продуктивність розробника, а й знижує ймовірність помилок, зокрема пов’язаних із нульовими посиланнями (null reference). Мова забезпечує безпечну роботу з null-значеннями завдяки системі nullable/non-nullable типів, яка реалізується вже на етапі компіляції.

Kotlin поєднує в собі функціональні та об'єктно-орієнтовані парадигми програмування, що надає розробникам гнучкість у виборі стилю реалізації програмної логіки. Зокрема, мова підтримує вищі функції, lambda-вирази,

розширення класів, деструктуризацію об'єктів, а також data-класи, що значно спрощує роботу з моделями даних у мобільному застосунку.

Для створення Android-застосунку на Kotlin використовується той самий набір інструментів, що й для Java, включаючи Android SDK, Android Jetpack, бібліотеки Room, Retrofit, LiveData, ViewModel тощо. Завдяки повній сумісності з Java-інфраструктурою, Kotlin може взаємодіяти з існуючими бібліотеками без додаткових обмежень. У межах одного проєкту можлива одночасна робота з кодом, написаним як на Java, так і на Kotlin, що полегшує міграцію старих проєктів або поступове переписування модулів.

Значну увагу в Kotlin приділено безпеці та стабільності коду. Наприклад, використання операторів “?” та “!!” дозволяє розробникові чітко визначати, які змінні можуть бути null, і як вони повинні оброблятися. Це знижує ризик помилок під час виконання програми. Крім того, Kotlin має ефективну систему обробки виключень, що забезпечує контрольоване реагування застосунку на помилки.

У поєднанні з архітектурними компонентами Android Jetpack, Kotlin демонструє високу ефективність у розробці застосунків, які дотримуються сучасних підходів, таких як MVVM (Model-View-ViewModel), Clean Architecture або MVI. Мова також активно використовується у фреймворках для реактивного програмування (наприклад, Kotlin Coroutines), що дозволяє зручно працювати з асинхронними запитами, потоками даних і паралельними обчисленнями.

На рівні екосистеми Kotlin має потужну підтримку від спільноти та розробників Android. Постійні оновлення мови та офіційна документація сприяють швидкому освоєнню інструментів та технологій навіть початківцями. Багато навчальних ресурсів, зокрема офіційні посібники Google, тепер публікуються з використанням саме Kotlin, що підкреслює актуальність мови в сучасній Android-розробці.

Таким чином, Kotlin є сучасним, безпечним і продуктивним вибором для розробки мобільних застосунків на платформі Android. Його синтаксична лаконічність, підтримка сучасних парадигм програмування та тісна інтеграція з

Android-інструментами роблять мову надзвичайно привабливою як для нових проєктів, так і для поступової модернізації вже існуючих рішень.

Порівняння швидкодії цих мов програмування у різних ситуаціях наведено в таблиці 2.2 [17].

2.2. Порівняння швидкодії Java та Kotlin у різних ситуаціях

Тест-кейс	Java	Kotlin
Ітерація циклу	Швидше	Повільніше
Створення об’єктів	Повільніше	Швидше
Виклик методів	Швидше	Повільніше

Проаналізувавши переваги Java та Kotlin, було прийнято рішення використовувати мову програмування Java для розробки модулів мобільного застосунку «DronePathfinder».

Однією з ключових функцій багатьох застосунків є відображення інформації на мапі, що дозволяє користувачам зручно навігуватися, знаходити місця та отримувати необхідну географічну інформацію. Вибір оптимального засобу для відображення мапи в Android-застосунку є важливим рішенням, яке впливає на функціональність, продуктивність та задоволення користувачів. Двома найпопулярнішими сервісами для інтеграції карт є Google Maps API та OpenStreetMap.

Google Maps API (Application Programming Interface) є комплексом інструментів і сервісів, які надає компанія Google для інтеграції картографічних можливостей до веб- або мобільних застосунків. Цей інтерфейс програмування застосунків дозволяє розробникам ефективно працювати з географічними об’єктами, створювати динамічні карти, здійснювати побудову маршрутів, розрахунок відстаней, геокодування адрес, а також надавати інтерактивні сервіси на основі місцезнаходження.

Google Maps API тісно інтегрується з Android-платформою через Google Maps SDK for Android, що забезпечує доступ до всіх основних картографічних функцій безпосередньо з мобільного пристрою. Завдяки цьому розробники мають змогу відображати карти без потреби у зовнішніх браузерах або сторонніх

сервісах. Основними можливостями SDK є підтримка масштабування, панорамування, накладення маркерів, полігонів, ліній та інформаційних вікон, а також обробка користувацьких подій.

Однією з найважливіших функціональних можливостей API є побудова маршрутів за допомогою сервісу Directions API, який дозволяє прокладати оптимальні шляхи між двома або більше точками з урахуванням типу транспорту (пішки, автомобілем, велосипедом або громадським транспортом). Розрахунок маршруту супроводжується детальною інформацією щодо відстані, тривалості подорожі, а також покроковою навігацією. Для мобільних застосунків це є критично важливим з погляду реалізації логіки переміщення, логістики або доставки [18].

OpenStreetMap (OSM) — це відкрита й колективно редагована картографічна платформа, яка забезпечує вільний доступ до географічної інформації будь-якого масштабу. Проєкт був започаткований у 2004 році Стівом Костом (Steve Coast) у відповідь на обмеження, що накладалися на доступ до картографічних даних у багатьох країнах. Сьогодні OpenStreetMap є однією з найбільших баз просторових даних у світі, яка постійно оновлюється спільнотою добровольців, професійних картографів, урядових і комерційних організацій.

На відміну від комерційних картографічних сервісів (таких як Google Maps або Here Maps), OSM не лише надає користувачеві доступ до карти, але й дозволяє вільно завантажувати, використовувати, змінювати та поширювати картографічні дані без ліцензійних обмежень, що особливо важливо для досліджень, розробки відкритих програмних продуктів та стартапів. Дані у проєкті публікуються під ліцензією Open Database License (ODbL), яка гарантує свободу використання з обов’язковим зазначенням джерела.

Основу структури OSM становлять векторні геодані, представлені у вигляді точок (nodes), ліній (ways) та відношень (relations), що разом формують як прості об’єкти (наприклад, будівлі чи дороги), так і складні просторові структури (наприклад, транспортні мережі чи межі адміністративних одиниць). До кожного об'єкта додається набір атрибутів у форматі ключ-значення (tags), що забезпечує

семантичне наповнення геоданих і дозволяє здійснювати пошук, фільтрацію й аналітику за допомогою геоінформаційних систем.

OpenStreetMap активно використовується в мобільній розробці, особливо у проєктах, де потрібне автономне функціонування, робота в режимі офлайн, або повна контрольованість над картою. Існує велика кількість інструментів і бібліотек, що дозволяють інтегрувати OSM у мобільні застосунки на Android, зокрема: OSMDroid, Mapsforge, Mapbox GL, Leaflet (у гібридних застосунках), а також повноцінні SDK, які дозволяють працювати з рендерингом, навігацією та маршрутизацією.

Однією з ключових переваг OpenStreetMap є можливість адаптації даних до конкретних цілей застосунку. Наприклад, можливо створити власну стилізацію карти, обмежити типи об’єктів, що відображаються, або реалізувати спеціалізовані маршрути — для пішоходів, дронів, велосипедистів, тощо. Також, за допомогою зовнішніх сервісів, таких як GraphHopper, OSRM (Open Source Routing Machine) або Valhalla, розробники можуть реалізовувати маршрутизацію на основі OSM-даних із підтримкою алгоритмів пошуку найкоротшого шляху (наприклад, Дейкстри або A*) [19].

Враховуючи особливості Google Maps API та OpenStreetMap було прийнято рішення використовувати OpenStreetMap при розробці модуля відображення мапи у застосунку.

РОЗДІЛ 3 РОЗРОБКА СТРУКТУРИ ТА ПРОГРАМНИХ МОДУЛІВ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Розробка структури та інтерфейсу мобільного застосунку

Структурна схема - це візуальне представлення, яке окреслює ключові компоненти програмного продукту, їхні зв'язки та функції. Ця схема може включати елементи, такі як підсистеми, модулі, бібліотеки та інші ресурси, які спільно працюють для досягнення загальної мети програми.

Процес створення структурної схеми часто вимагає ітеративного підходу, де кожен крок додає більше деталей до загальної картини, дозволяючи розробникам краще розуміти та оптимізувати взаємодію між різними частинами програми [20].

Структурна схема застосунку «DronePathfinder» зображено на рисунку 3.1.

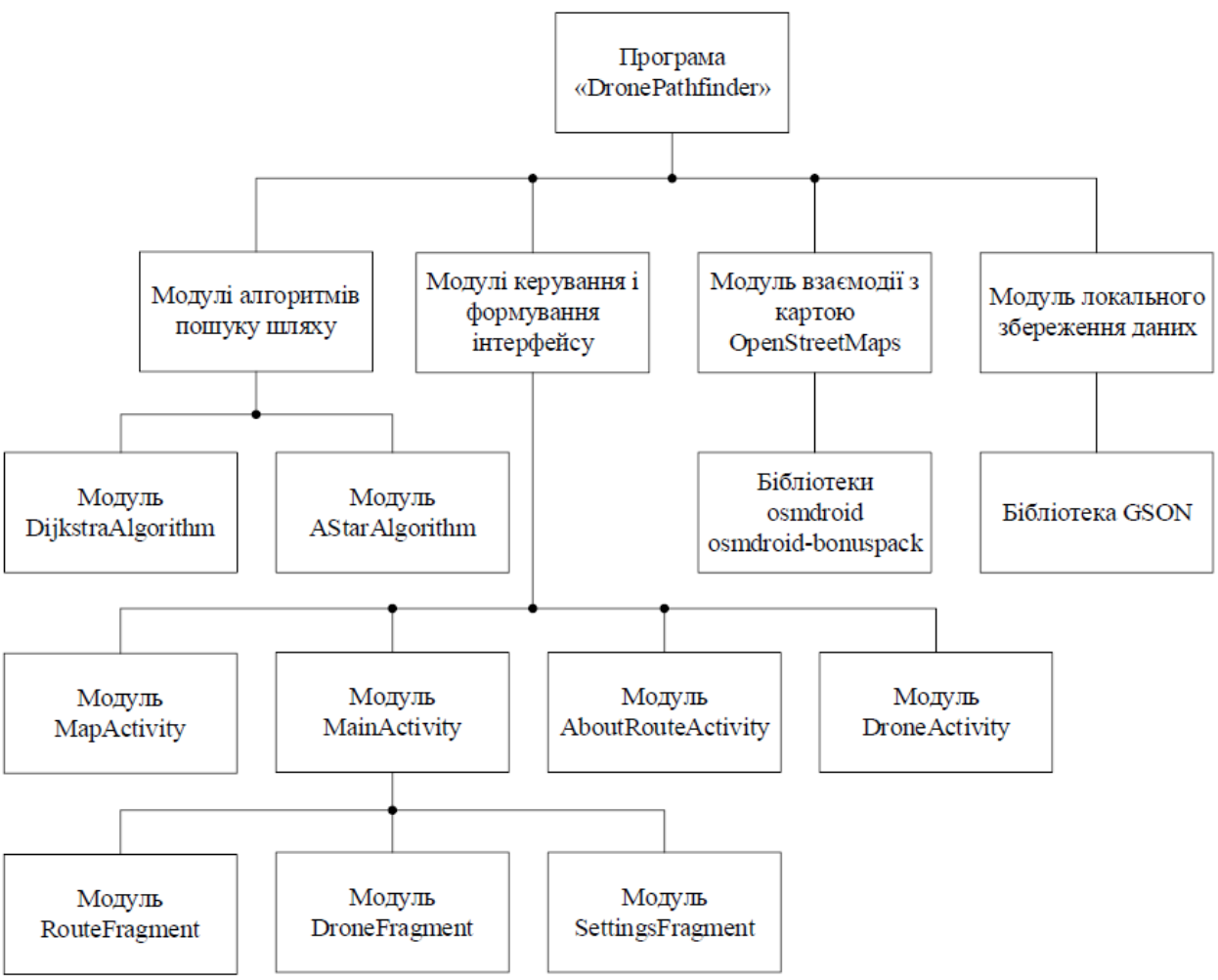


Рис. 3.1. Структурна схема застосунку «DronePathfinder»

Інтерфейс користувача складається з чотирьох активностей та трьох фрагментів, що входять у головну активність. До активностей застосунку відносяться:

− MainActivity – використовується для відображення головного розділу;

− MapActivity – використовується для відображення розділу планування маршруту;

− AboutRouteActivity – використовується для відображення розділу «Про маршрут»;

− DroneActivity – використовується для відображення розділу «Дрон».

На рисунку 3.2 зображено графічну схему головної активності застосунку «DronePathfinder».

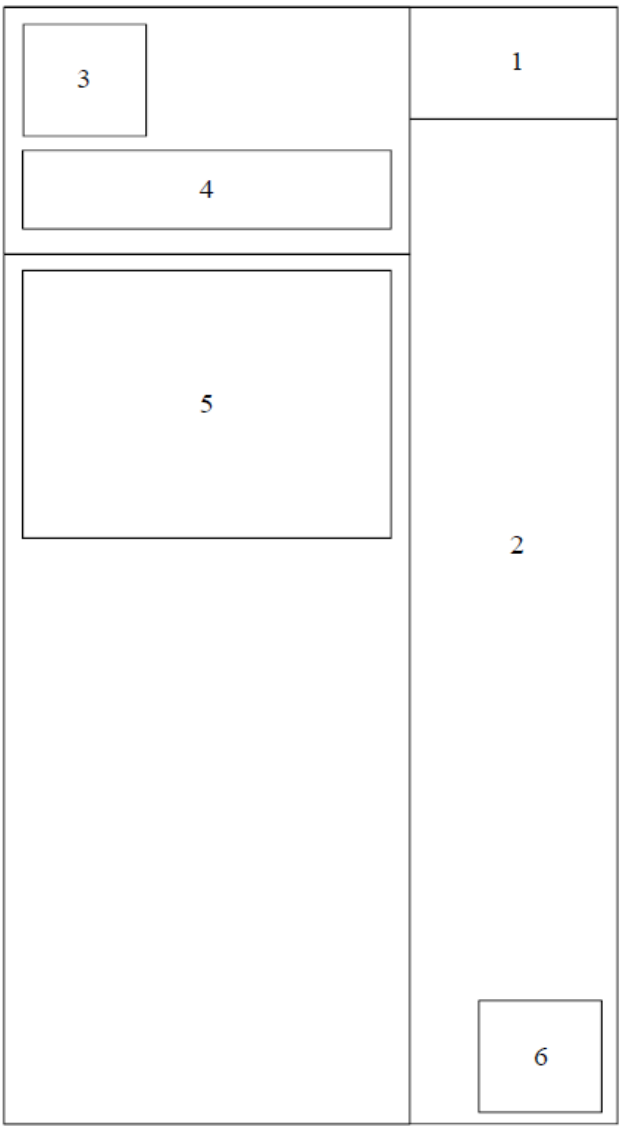


Рис. 3.2. Графічна схема головної активності застосунку «DronePathfinder»

До елементів головної активності належать:

1. Заголовок активності.

2. Робоча область активності.

3. Логотип мобільного застосунку «DronePathfinder».

4. Інформація про розробника мобільного застосунку «DronePathfinder».

5. Пункти навігаційного меню – містить пункти «Маршрути», «Безпілотники», «Налаштування».

6. Основна кнопка взаємодії з фрагментом.

Основними кольорами мобільного застосунку було обрано синій та блакитний.

Розробку графічного інтерфейсу користувача було проведено у середовищі Android Studio з використанням вбудованого редактора.

Інсталяційний файл мобільного застосунку поширюється у форматі .APK файлу.

Для інсталяції застосунку «DronePathfinder» користувачу необхідно виконати такі дії:

1. Завантажити файл DronePathfinder.apk на свій мобільний пристрій.

2. Перейти до «Налаштувань» на пристрої.

3. Обрати розділ «Безпека» або «Приватність».

4. Знайти опцію «Встановлення з невідомих джерел» і активувати її.

5. Відкрити файловий менеджер на пристрої.

6. Знайти завантажений APK-файл.

7. Натиснути на файл, щоб почати встановлення. У випадку якщо система запитає дозвіл на встановлення застосунку, користувачу необхідно підтвердити свій вибір.

8. Дочекатись повідомлення про успішне встановлення.

Після завершення встановлення користувач отримує змогу відкрити застосунок через меню додатків або піктограму на головному екрані.

Щоб видалити застосунок з мобільного пристрою, користувачу необхідно виконати такі дії:

1. Відкрити застосунок «Налаштування» на пристрої.

2. Прокрутити до розділу «Додатки» або «Менеджер додатків».

3. Знайти та обрати застосунок «DronePathfinder».

4. Натиснути на кнопку «Видалити» (рисунок 3.3).

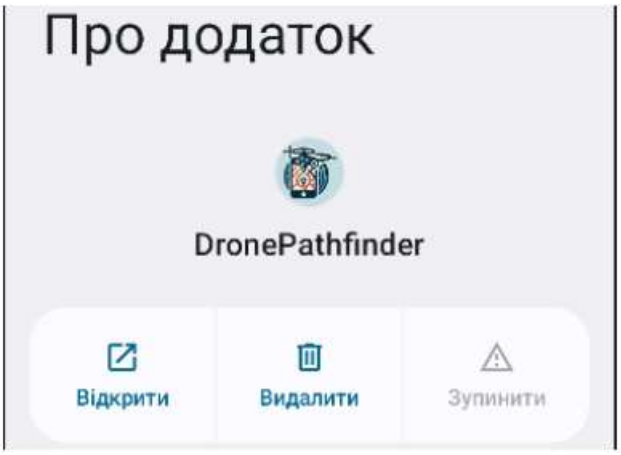


Рис. 3.3. Сторінка «Про додаток»

5. Необхідно підтвердити вибір, якщо система запитає про це (рисунок 3.4).



Рис. 3.4. Вікно підтведження видалення застосунку

Після видалення застосунку видаляються усі дані застосунку, в тому числі маршрути, профілі безпілотників та налаштування застосунку.

При першому відкритті застосунку користувача зустрічає екран завантаження з логотипом (рисунок 3.5).



Рис. 3.5. Екран завантаження застосунку

Після завантаження застосунку відкривається розділ «Маршрути» у якому користувач може керувати переліком своїх маршрутів. При першому відкритті застосунку розділ «Маршрути» є порожнім (рисунок 3.6).

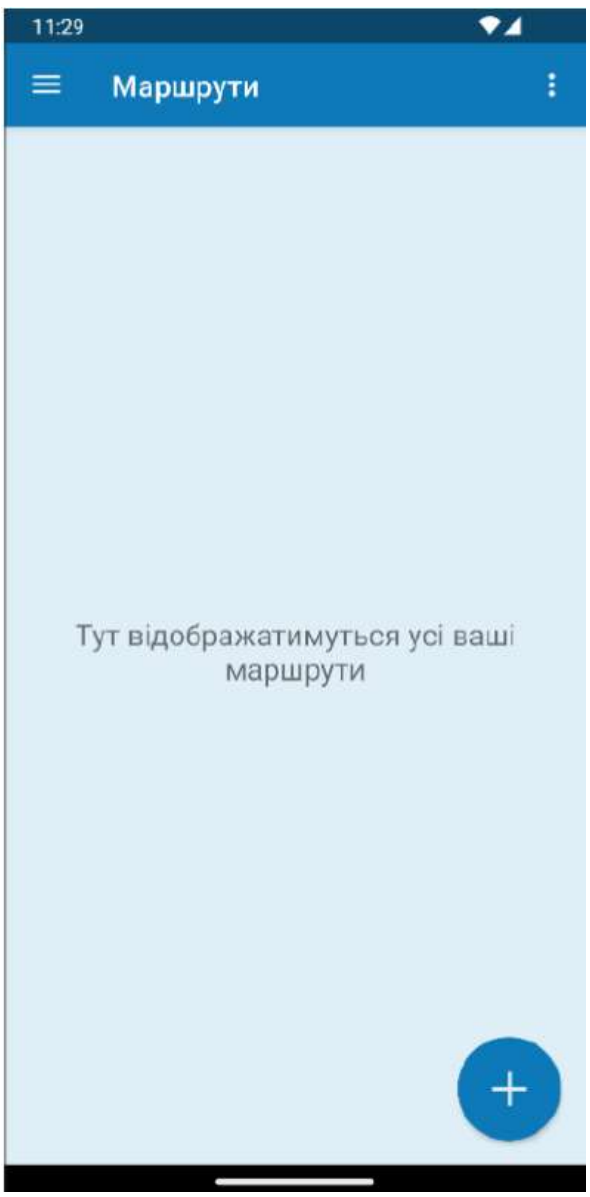


Рис. 3.6. Розділ «Маршрути»

Щоб створити новий маршрут користувачу необхідно натиснути на кнопку «+» в правому нижньому куті екрану. Після натиснення кнопки відкривається розділ «Планування маршруту» з порожньою мапою (рисунок 3.7). Для масштабування мапи користувач може використати мультітач жести.

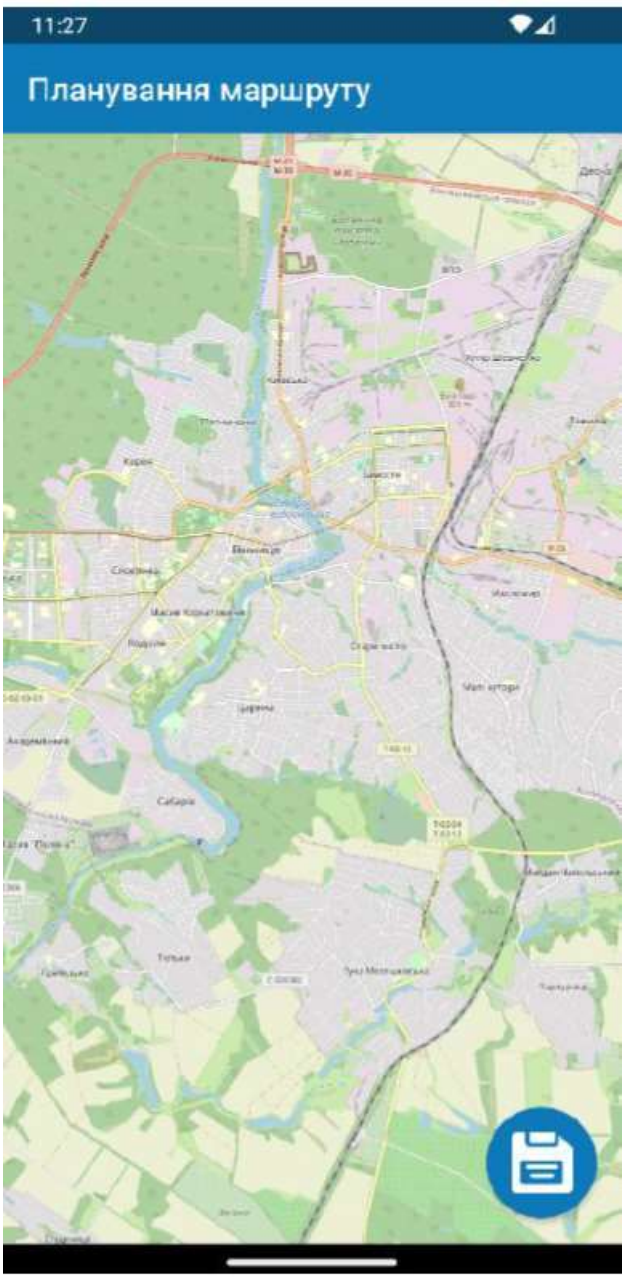


Рис. 3.7. Розділ «Планування маршруту»

Щоб додати нову мітку на мапу і розпочати процес створення маршруту користувачу необхідно натиснути у будь-якому місці на мапу у розділі «Планування маршруту» (рисунок 3.8).

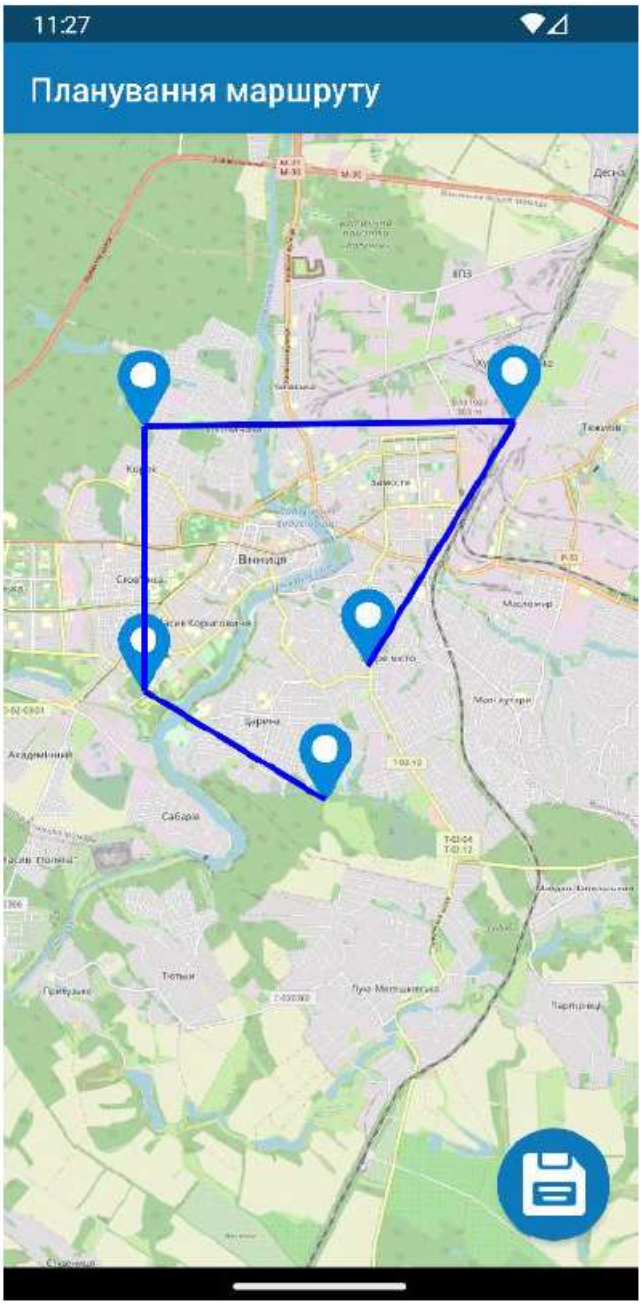


Рис. 3.8. Додання міток на мапу у розділі «Планування маршруту»

Щоб видалити мітку з мапи користувачу необхідно повторно натиснути на існуючу мітку на мапі у розділі «Планування маршруту» (рисунок 3.9).

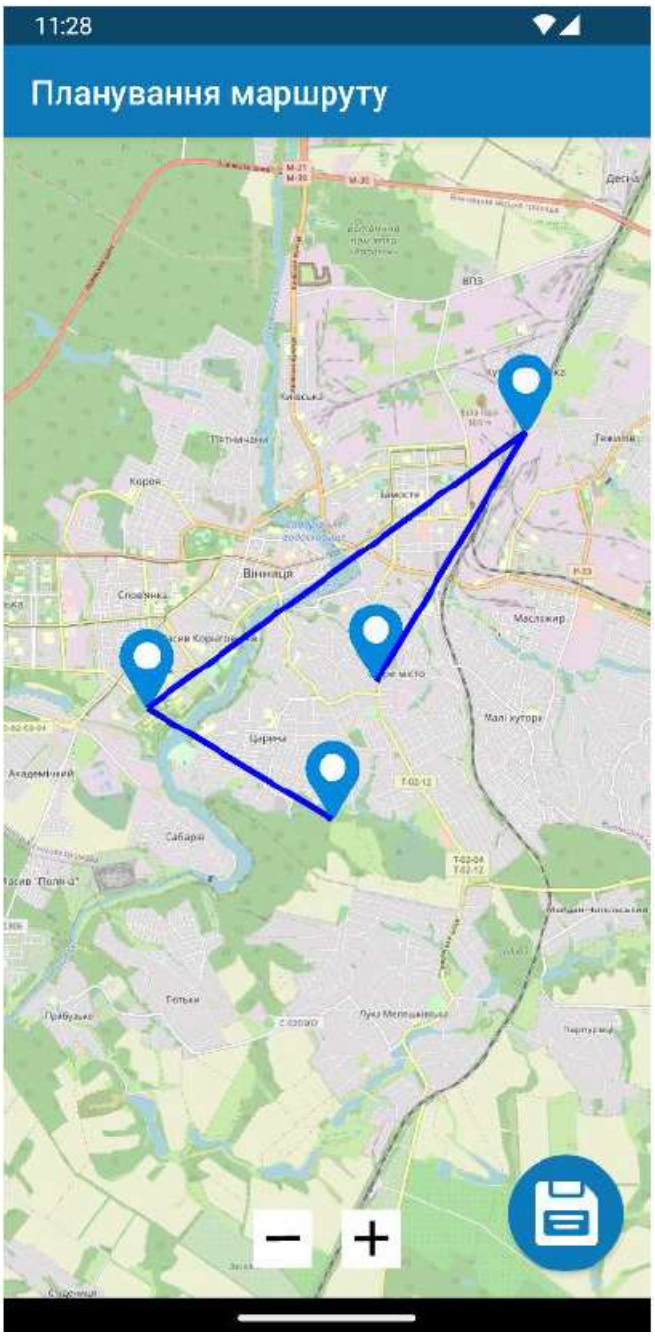


Рис. 3.9. Видалення міток з мапи у розділі «Планування маршруту»

Для того щоб додати точку запобігання, яка буде обмежувати точки, що використовуються алгоритмом для знаходження оптимального шляху, користувачу необхідно використати жест «натиснути і утримати» у будь-якому місці на мапі. Після виконання цього жесту відкриється вікно (рисунок 3.10), у якому необхідно ввести радіус точки запобігання у метрах.

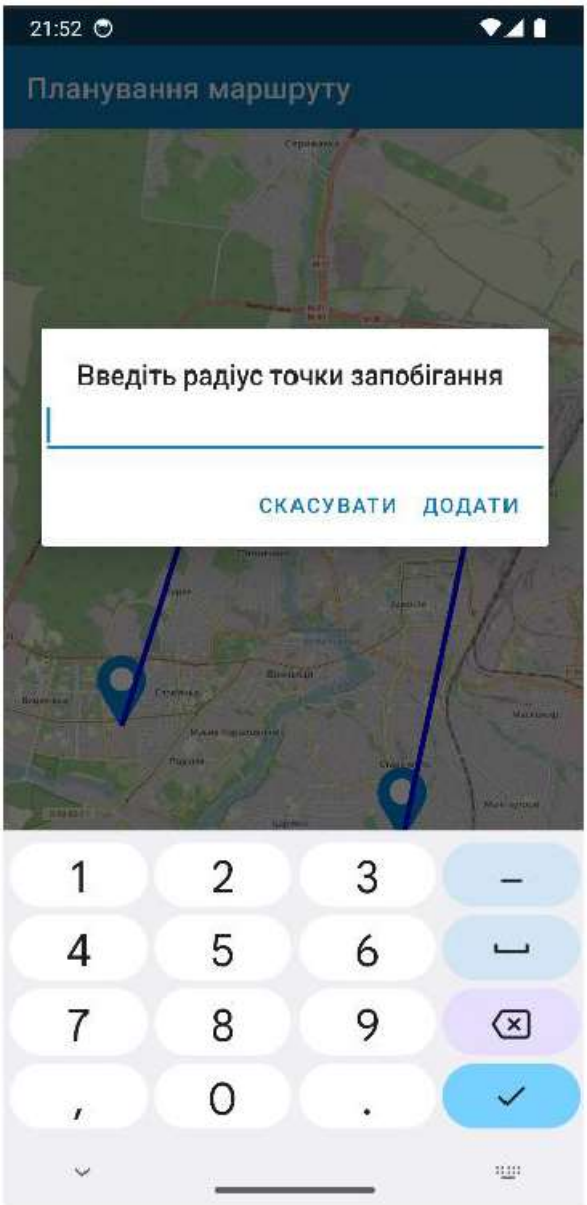


Рис. 3.10. Вікно додання точки запобігання у розділі «Планування маршруту»

Після підтвердження операції додання точки запобігання натисканням кнопку «Додати» на мапі з’являється коло червоного кольору, що відображає додану зону (рисунок 3.11).



Рис. 3.11. Додана точка запобігання на мапі

Після завершення процесу планування маршруту користувач повинен натиснути кнопку з піктограмою дискети для його збереження. Користувача автоматично повертає у розділ «Маршрути» і відображає у кінці списку збережений маршрут (рисунок 3.12).

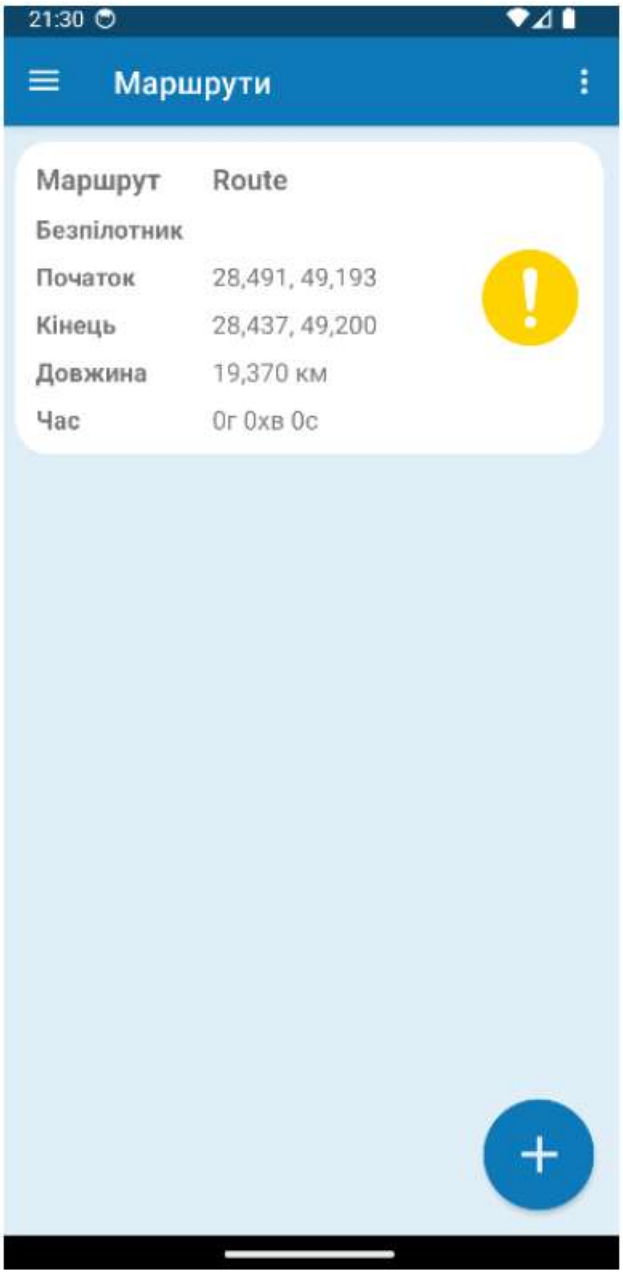


Рис. 4.12. Збережений маршрут у розділі «Маршрути»

У випадку, якщо до маршруту не додано безпілотник, відображається статус з жовтою піктограмою знака оклику, інакше, якщо його вантажопідйомність та дальність польоту більша за вимоги маршруту – зеленою піктограмою «Checkmark», якщо ж якийсь із параметрів менший – червона піктограма (рисунок 3.13).

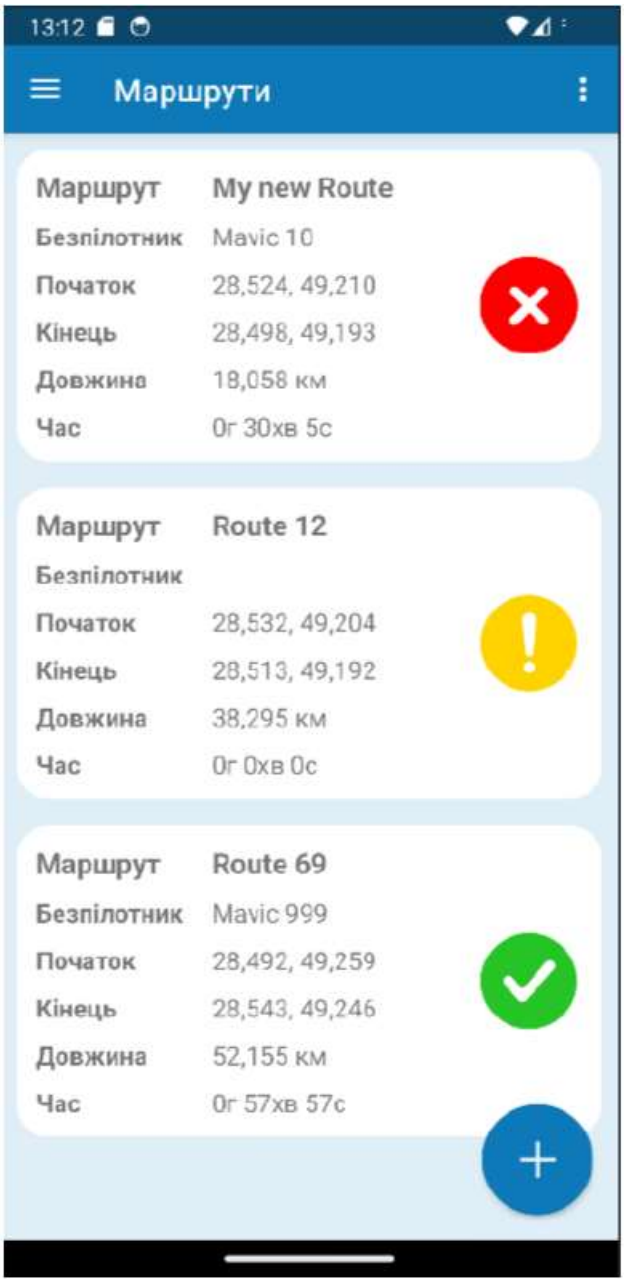


Рис.3.13. Відображення статусу маршруту у списку

Інтерфейс та локалізація застосунку автоматично адаптується до мови системи: українська у випадку використання української мови системи, у всіх інших випадках – англійська (рисунок 3.14).

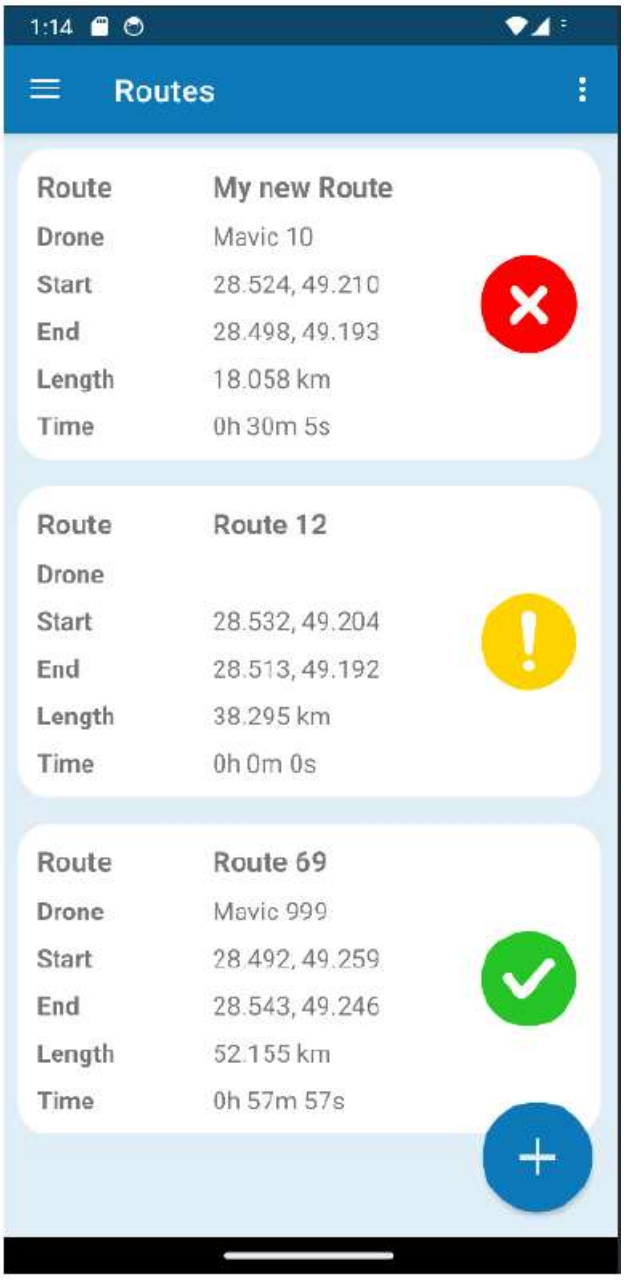


Рис. 3.14. Зміна мови застосунку

Щоб отримати можливість переглянути більше інформації про маршрут, змінити безпілотник, що використовується у маршруті, або видалити маршрут, користувачу необхідно натиснути на маршрут у списку. Після чого відкривається активність «Про маршрут» (рисунок 3.15). Також у цій активності користувач може переглянути статус та рекомендації щодо маршруту.

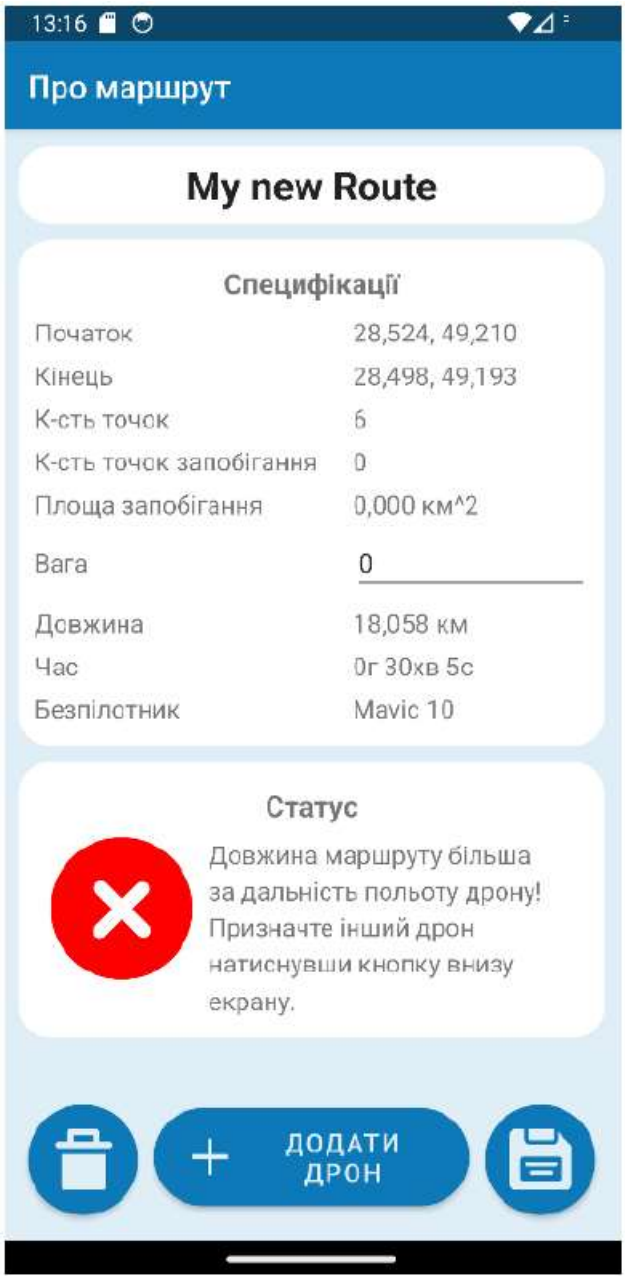


Рис.3.15. Активність «Про маршрут»

При натисканні на кнопку «Додати дрон» з’являєтся діалогове вікно вибору безпілотників (рисунок 3.16), у якому користувач може обрати який саме безпілотник додати до маршруту.

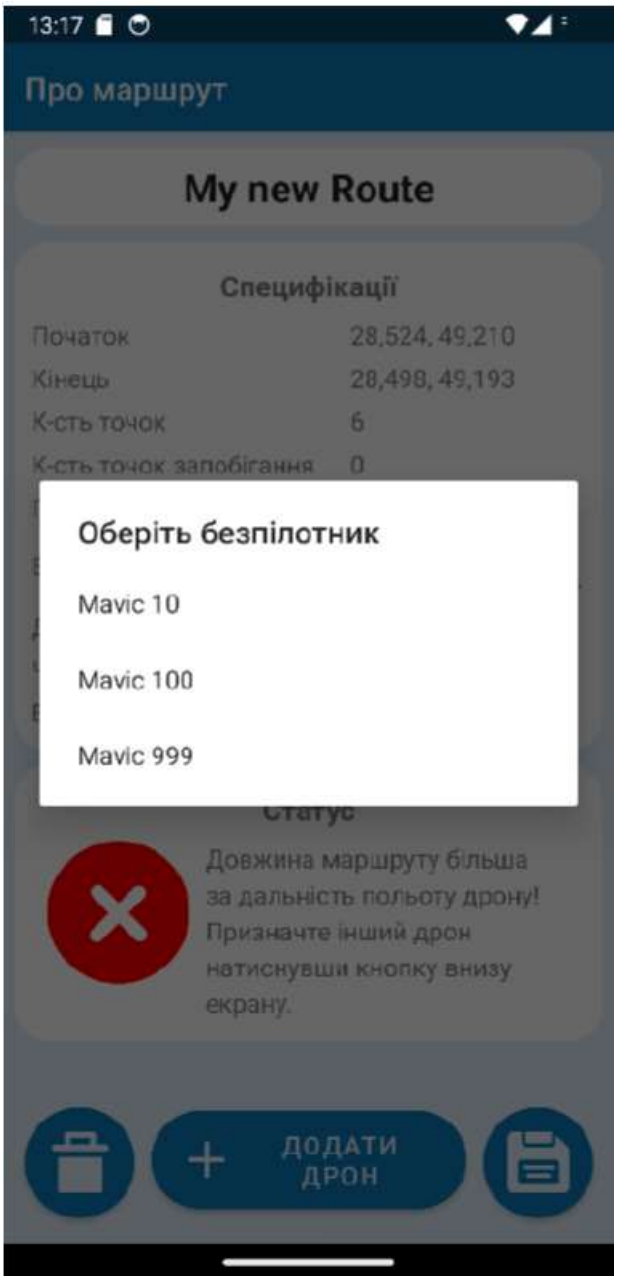


Рис. 3.16. Додання безпілотника до маршруту

Для видалення маршруту користувачу достатньо натиснути на кнопку з відповідною піктограмою і підтвердити операцію видалення у діалоговому вікні що з’явиться (рисунок 3.17).

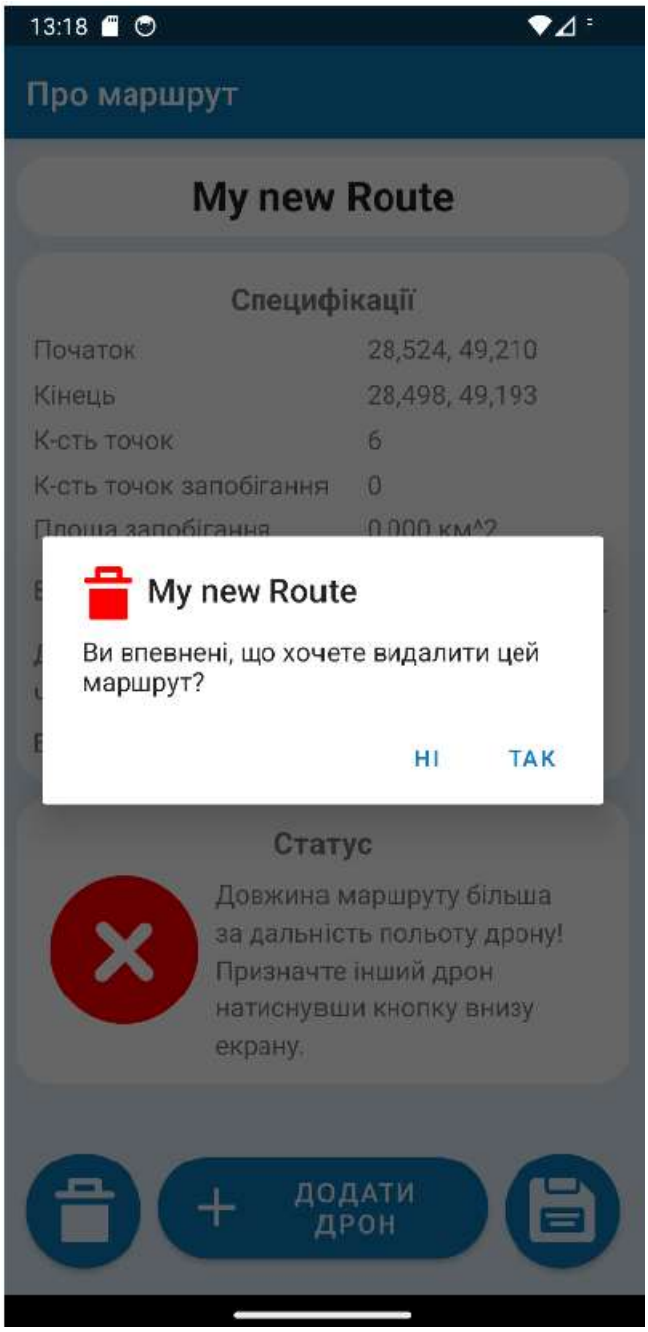


Рис. 3.17. Вікно підтвердження видалення маршруту

Для перегляду та керування списком безпілотників використовується активність «Безпілотники». Щоб додати новий дрон користувачу необхідно натиснути на кнопку з піктограмою «+» в правій нижній частині активності (рисунок 4.18).



Рис. 3.18. Активність «Безпілотники»

Щоб отримати можливість переглянути більше інформації про безпілотник, змінити його характеристики або видалити, користувачу необхідно натиснути на дрон у списку. Після чого відкривається активність «Безпілотник» (рисунок 3.19).

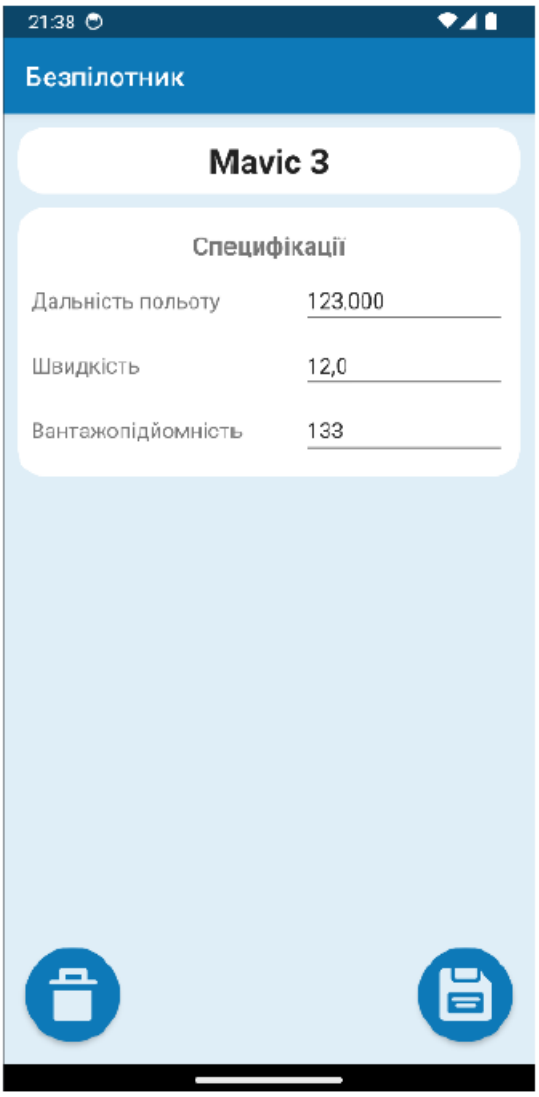


Рис. 3.19. Можливість перегляду та зміни характеристик дрона

В активності «Безпілотник» користувач може змінити таку інформацію про дрон як:

– назва;

– дальність польоту;

– швидкість;

– вантажопідйомність

Для збереження змін користувачу необхідно натиснути на кнопку з піктограмою дискети. У випадку, якщо хоча б одне поле залишиться порожнім, користувач отримає відповідне повідомлення (рисунок 3.20)

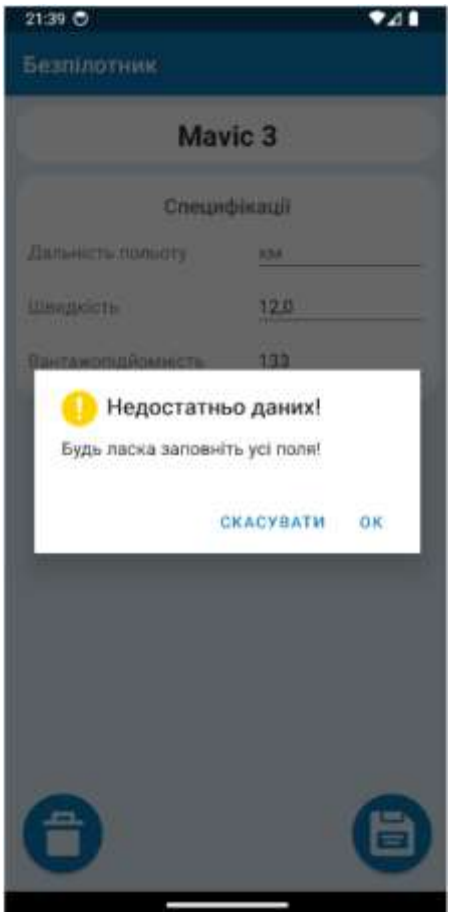


Рис. 3.20. Повідомлення про відсутність даних

Однією з головних функціональних характеристики застосунку є можливість вибору алгоритму знаходження оптимального шляху (рисунок 4.21).

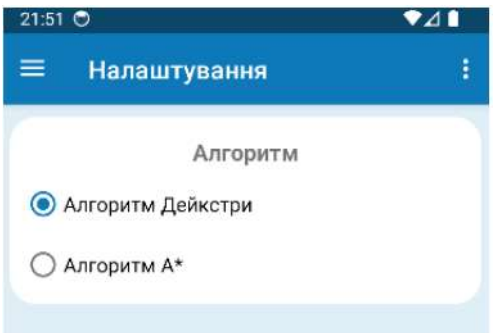


Рис. 3.21. Вибір алгоритму в налаштуваннях

Щоб змінити алгоритм користувачу необхідно в активності «Налаштування» обрати бажаний алгоритм.

3.2 Використання принципів об’єктно-орієнтованого програмування при розробці мобільного застосунку

Для уникнення повторюваності коду використано один із найважливіших принципів об’єктно-орієнтованого програмування – наслідування. Враховуючи що класи Route та Drone міститимуть однакові поля: назва, статус; для цих класів

було створено батьківський клас Object, який містить ці поля. При цьому клас Object є абстрактним, що унеможливлює створення об’єктів саме цього класу.

Використання іншого не менш важливого принципа об’єктно-орієнтованого програмування – інкапсуляції – реалізовано таким чином: усі поля класів Route, Drone, Object та AvoidancePoint мають модифікатор доступу private, а доступ до полів здійснюється з використанням мутаторів доступу.

У взаємозв’язку створених класів використано агрегацію: кожен об’єкт класу Route може містити об’єкт класу Drone та масив об’єктів AvoidancePoint.

AvoidancePoint – клас створений для зберігання інформації про точки запобігання – центр, що описується двома координатами, та його радіус.

На рисунку 3.22 зображено діаграму класів об’єктів призначених для зберігання мобільного застосунку.

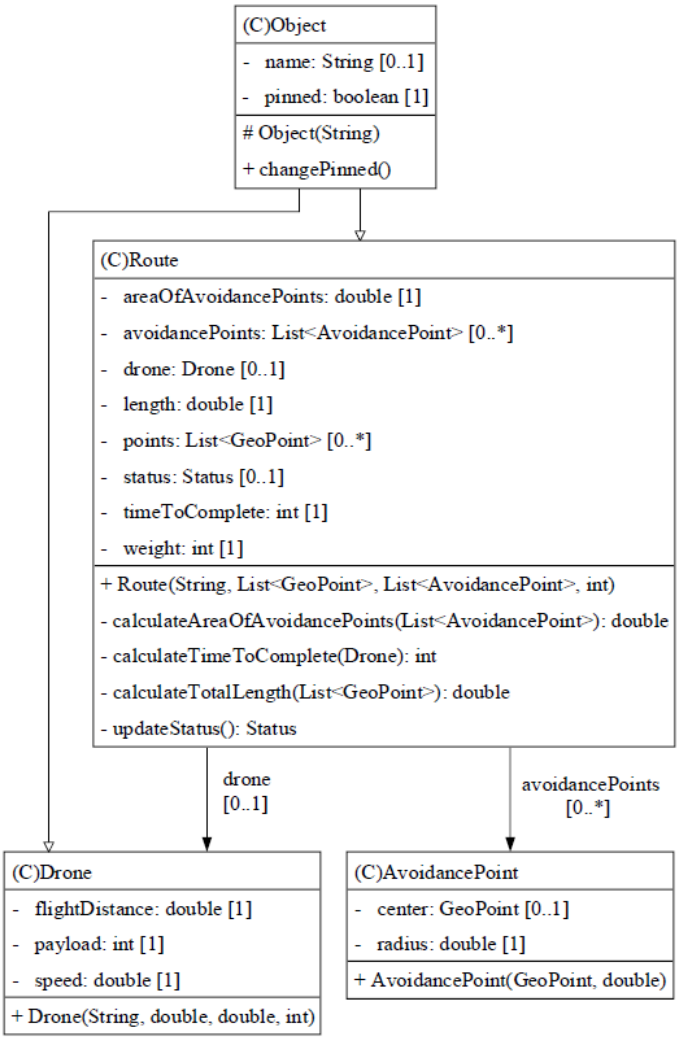


Рис. 3.22. Діаграма класів мобільного застосунку

Класи Route, Drone, Object та AvoidancePoint імплементують інтерфейс Serializable, що надає можливість виконувати серіалізацію/десереалізацію об’єктів цих класів.

3.3 Розробка модуля локального збереження даних

Основна задача модуля локального збереження даних полягає у керуванні списком маршрутів у мобільному додатку. Цей модуль є важливою частиною застосунку, оскільки він дозволяє користувачам переглядати, зберігати та відновлювати маршрути, які вони створили або використовували. Завдяки цьому, користувачі можуть легко повертатися до своїх улюблених або часто використовуваних маршрутів.

Метод saveRoutesToSharedPreferences() використовується для збереження даних про маршрути в локальній пам’яті пристрою за допомогою SharedPreferences (рисунок 3.23).

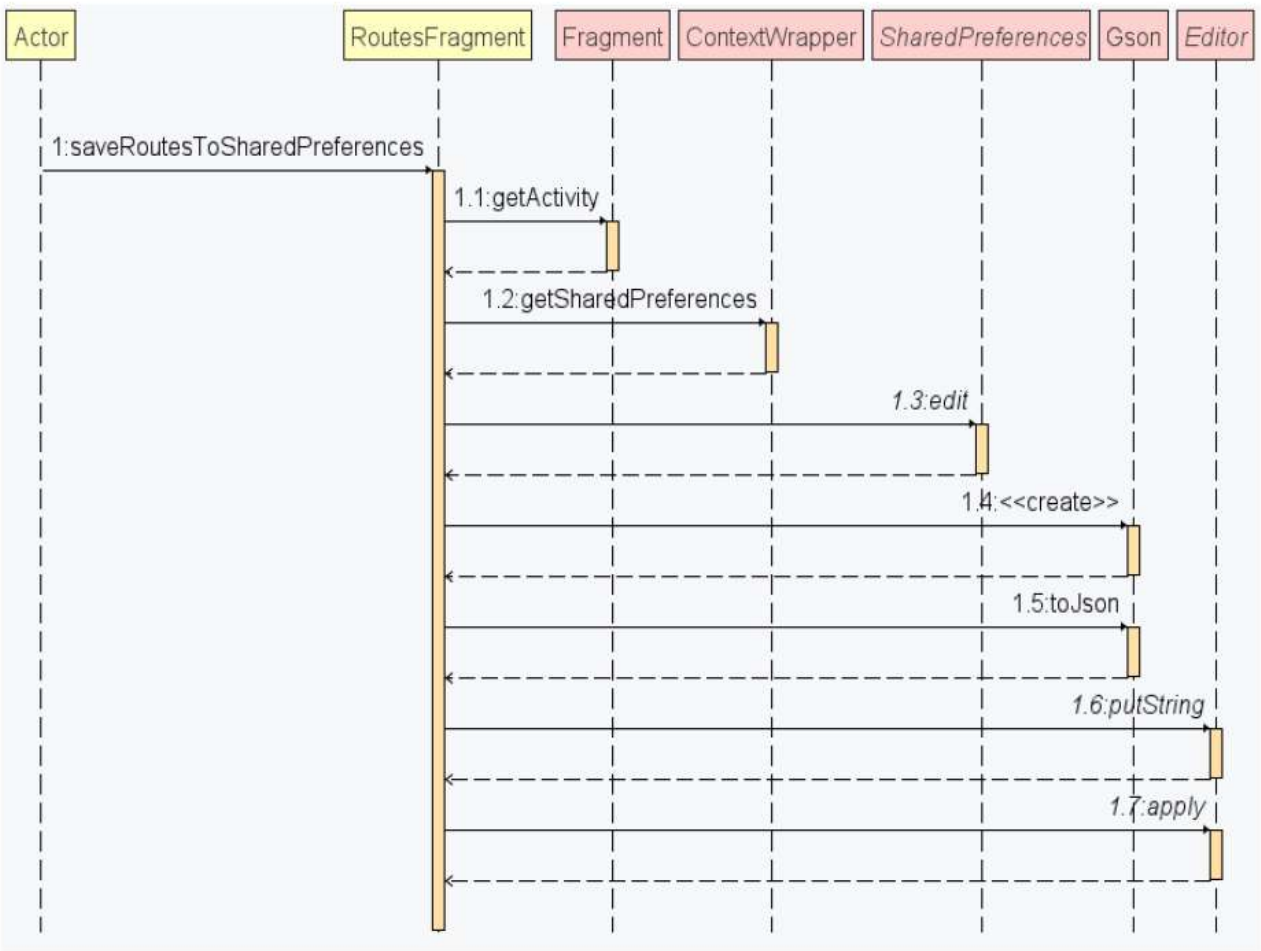


Рис.3.23. Діаграма послідовності методу saveRoutesToSharedPreferences

Він перетворює список маршрутів у формат JSON, а потім зберігає цей JSON рядок під певним ключем. Це дозволяє додатку зберігати стан маршрутів, щоб користувачі могли доступити до своїх маршрутів навіть після перезапуску додатку або пристрою. Такий підхід є стандартною практикою для збереження невеликих обсягів даних, які не вимагають постійного доступу до мережі або бази даних.

Метод loadRoutesFromSharedPreferences() відновлює збережений список маршрутів з локальної пам’яті пристрою (рисунок 3.24).

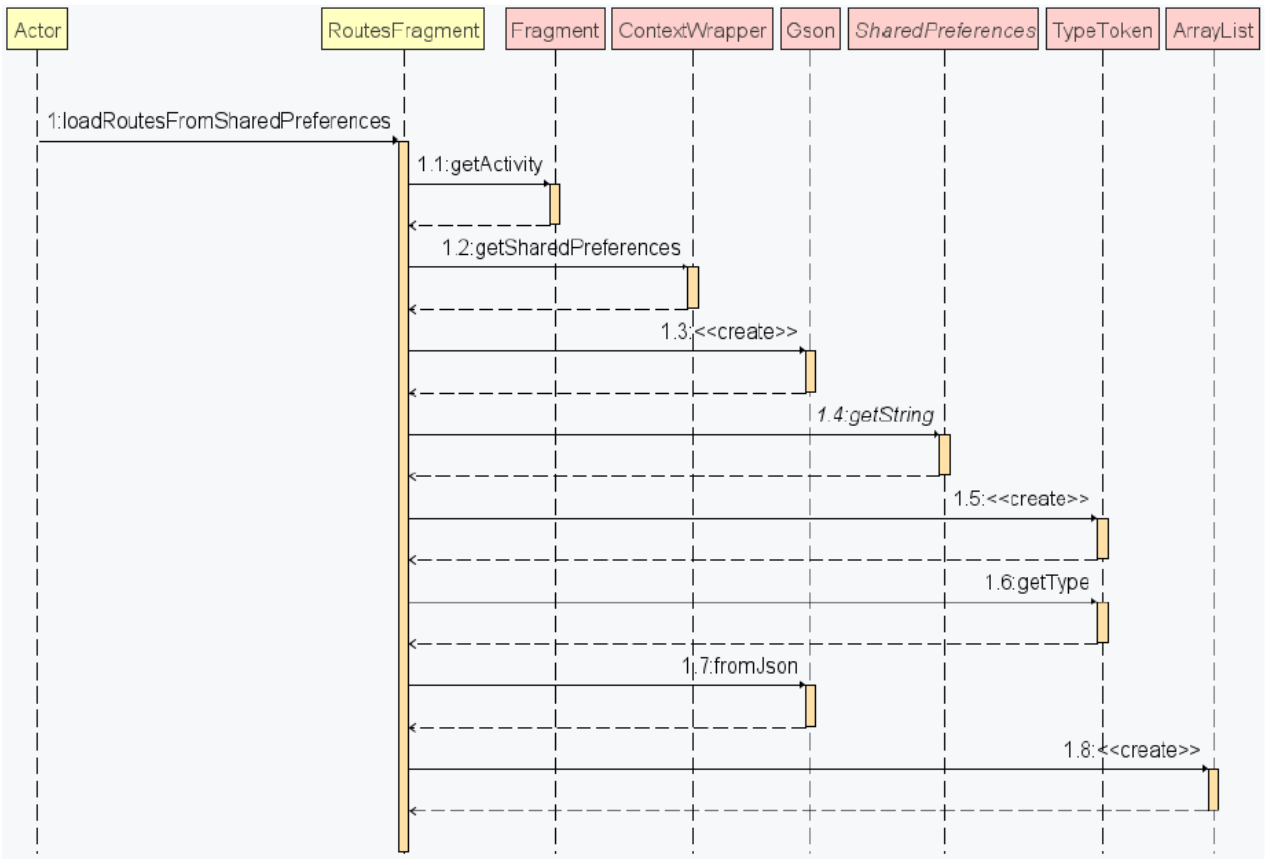


Рис. 3.24. Діаграма послідовності методу loadRoutesFromSharedPreferences

Він використовує SharedPreferences для отримання рядка у форматі JSON, який містить інформацію про маршрути. Потім цей рядок перетворюється назад у список об’єктів Route за допомогою бібліотеки Gson. Якщо збережені дані відсутні, метод створює новий порожній список. Таким чином, користувачі можуть продовжувати працювати зі своїми маршрутами після перезапуску додатку або пристрою.

Для керування списком маршрутів використовується адаптер, який дозволяє відображати дані про маршрути у вигляді списку на екрані.

Він перевизначає метод getView, щоб кожен маршрут у списку мав своє представлення. Коли користувач переглядає список, для кожного маршруту відображається інформація, яка включає назву маршруту, координати початкової та кінцевої точок, а також його довжину. Ця інформація форматується у читабельний текст, який потім встановлюється як текст для відображення у компоненті TextView.

3.4 Тестування мобільного застосунку

Для моніторингу продуктивності застосунку «DronePathfinder», логування подій та знаходження виключних ситуацій використовується Logcat (рисунок 3.25).

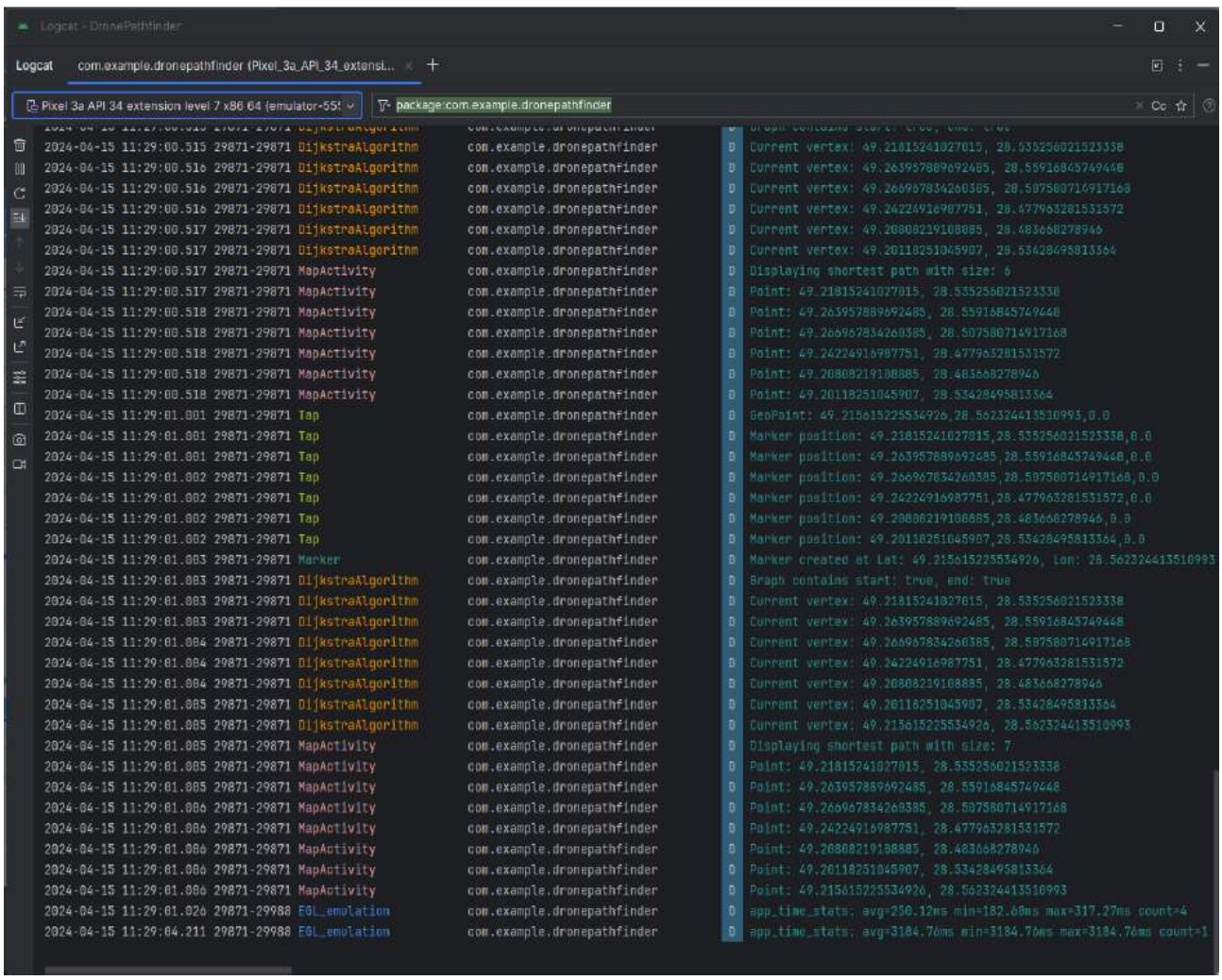


Рис. 3.25. Моніторинг з використанням Logcat

Logcat — це засіб логування, який виводить журнал системних повідомлень, при цьому можна додати свої повідомлення за допомогою класу Log [21]. Цей інструмент дозволяє відстежувати повідомлення, пов’язані з виконанням застосунка та операційної системи Android. Кожен лог містить дату, часову мітку, ідентифікатор процесу та потоку, тег, ім’я пакета, пріоритет та повідомлення. При виникненні виняткової ситуації Logcat показує повідомлення та пов’язаний стек викликів з посиланнями на рядки коду.

Юніт тести дуже важливі для розробки програмного забезпечення, оскільки вони забезпечують кілька ключових переваг [22]:

1. Юніт тести перевіряють, що окремі частини коду (модулі, класи, функції) працюють належним чином [22].

2. Вони допомагають виявити помилки та проблеми на ранніх етапах розробки, що може значно зменшити витрати на виправлення [22].

3. Написання юніт тестів може спонукати до кращого дизайну програми, оскільки вони вимагають, щоб код був легко тестованим [22].

4. Юніт тести можуть бути автоматизовані та виконуватися часто, що забезпечує постійну перевірку коду на наявність помилок [22].

Для перевірки роботи правильності алгоритму Дейкстри і виявлення помилок було створено юніт тести.

Результат виконання юніт тестів зображено на рисунку 3.26.

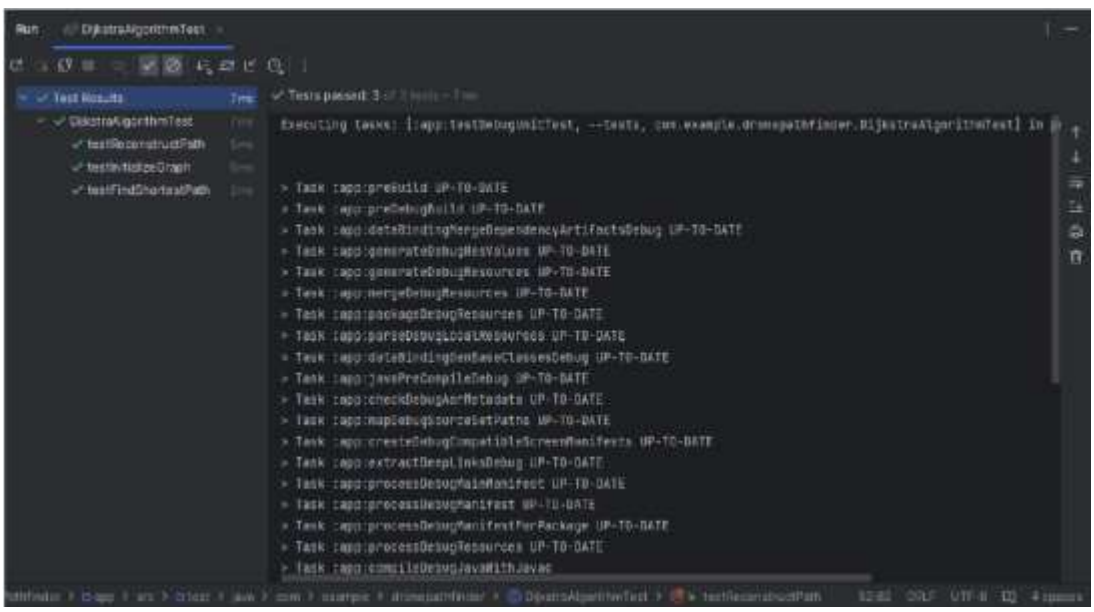


Рисунок 3.26.Результат виконання юніт тестів

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досліджено та реалізовано програмне рішення — мобільний застосунок для платформи Android, що дозволяє знаходити оптимальний маршрут для безпілотного літального апарату з урахуванням просторових обмежень та заданих критеріїв ефективності.

В результаті проведеного аналізу було встановлено, що одним з ключових викликів при плануванні маршрутів для БПЛА є необхідність врахування великої кількості змінних факторів: рельєфу місцевості, наявності перешкод, зон із забороною польотів, енергетичних обмежень та ін. Для розв'язання цього завдання були розглянуті та порівняні основні алгоритми маршрутизації, серед яких особлива увага приділялася алгоритму Дейкстри як одному з надійних методів пошуку найкоротшого шляху у графах із позитивними вагами.

В роботі було розроблено архітектуру мобільного застосунку «DronePathfinder», визначено його функціональні модулі та реалізовано користувацький інтерфейс. Застосунок використовує геоінформаційні дані та алгоритмічне ядро для побудови маршруту польоту з точки А в точку В з урахуванням заданих параметрів. Для візуалізації просторових даних та відображення маршрутів було використано API (Google Maps або OpenStreetMap), що дозволяє інтегрувати карти та маркери в реальному масштабі.

Проведене тестування підтвердило функціональну працездатність програмного забезпечення та правильність реалованих алгоритмів. Застосунок демонструє здатність адаптивно реагувати на зміну вхідних умов, ефективно обробляє просторові дані, будує маршрути з мінімальними затратами та забезпечує зручність у використанні для кінцевого користувача.

У процесі дослідження були досягнуті поставлені цілі: здійснено аналіз існуючих рішень, визначено вхідні та вихідні дані, реалізовано програмну модель планування маршруту та створено працюючий мобільний застосунок. Практична цінність роботи полягає в можливості подальшого використання цього застосунку для реальних польових задач у сфері дистанційного

зондування, моніторингу територій, доставки вантажів або пошуково-рятувальних операцій.

Перспективами подальшого розвитку є розширення функціональності додатку, зокрема: реалізація обліку погодних умов у режимі реального часу, підтримка 3D-карт і висот, інтеграція з камерами та сенсорами БПЛА, а також використання машинного навчання для адаптивного прогнозування маршрутів. Також перспективним є додавання підтримки декількох типів дронів із різними характеристиками та обмеженнями.

Таким чином, результати проведеного дослідження мають теоретичну і практичну значущість та можуть бути використані як основа для впровадження сучасних систем інтелектуального управління безпілотними літальними апаратами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Applications of Unmanned Aerial Vehicles | Encyclopedia MDPI. URL: https://encyclopedia.pub/entry/25512
2. Simple and Efficient Algorithm for Drone Path Planning | IEEE Conference Publication | IEEE Xplore URL: https://ieeexplore.ieee.org/document/9500370
3. Path Planning for Autonomous Drones: Challenges and Future Directions URL: https://www.mdpi.com/2504-446X/7/3/169
4. Матейко Є. В. Вплив алгоритмів знаходження оптимального шляху на ефективність безпілотного літального апарату / Є. В. Матейко, Г. О. Черноволик // Матеріали LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20696/17212
5. Drones and their Applications in Different Sectors | by Webnx | Medium URL: https://medium.com/@webnxoffice/drones-and-their-applications-in-different-sectors-d23f6e0fe087
6. Towards Fully Autonomous UAVs: A Survey URL: https://www.mdpi.com/1424-8220/21/18/6223
7. UAV Path Planning Optimization Strategy: Considerations of Urban Morphology, Microclimate, and Energy Efficiency Using Q-Learning Algorithm URL: https://www.mdpi.com/2504-446X/7/2/123
8. A Comprehensive Review of Recent Research Trends on Unmanned Aerial Vehicles (UAVs) URL: https://www.mdpi.com/2079-8954/11/8/400
9. PathPlanner – Software application for path planning for mobile systems - Fraunhofer IOSB URL: https://www.iosb.fraunhofer.de/en/projects-and-products/PathPlanner-Software_application_for_path_planning_for_mobile_systems.html
10. Software Overview - PX4 Autopilot URL: https://px4.io/software/software-overview/

11. Mission Planner Home — Mission Planner documentation URL: https://ardupilot.org/planner/
12. QGroundControl User Guide | QGC Guide (master) URL: https://docs.qgroundcontrol.com/master/en/qgc-user-guide/index.html
13. DJI GS Pro – DJI URL: https://www.dji.com/global/ground-station-pro
14. Інструкція, як будувати UML-діаграми | DOU URL: https://dou.ua/forums/topic/40575/
15. Kotlin VS Java – What's the Difference? URL: https://www.freecodecamp.org/news/kotlin-vs-java-whats-the-difference/
16. Java Vs Kotlin – Javatpoint URL: https://www.javatpoint.com/java-vs-kotlin
17. Kotlin vs. Java: Which One is Better? – TechMagic URL: https://www.techmagic.co/blog/kotlin-vs-java/
18. Google Maps Platform Documentation | Maps SDK for Android | Google for Developers URL: https://developers.google.com/maps/documentation/android-sdk
19. osmdroid/osmdroid: OpenStreetMap-Tools for Android URL: https://github.com/osmdroid/osmdroid
20. Структурна схема URL:https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D0%B0
21. logcat | Android Developers URL: https://stuff.mit.edu/afs/sipb/project/android/docs/tools/help/logcat.html
22. 5 Benefits of Unit Testing and Why You Should Care URL: https://cswsolutions.com/blog/posts/2022/december/5-benefits-of-unit-testing-and-why-you-should-care/
23. Сидоренко В.А. Безпілотні літальні апарати: конструкція та управління. К.: Либідь, 2020. — 280 с.
24. Паламарчук О.В. Методи оптимізації маршрутів у задачах планування руху. Київ: КНУ, 2019. 220 с.

25. Лисенко О.П. Інтелектуальні системи керування: основи теорії та практики. Харків: ХНУ, 2018. 312 с.
26. Козлов А.В. Автоматизація навігаційних систем безпілотних літальних апаратів. Дніпро: ДНУ, 2020. 198 с.
27. Головко О.М. Теорія оптимального керування. К.: НАУ, 2020. 340 с.
28. Siegwart R., Nourbakhsh I.R., Scaramuzza D. Introduction to Autonomous Mobile Robots. MIT Press, 2011.
29. Kochenderfer M.J. Decision Making Under Uncertainty: Theory and Application. MIT Press, 2015.
30. Sutton R.S., Barto A.G. Reinforcement Learning: An Introduction. — MIT Press, 2018.
31. Yang K., et al. Survey of Autonomous UAV Path Planning. IEEE Access, 2020, 8: 45684-45701.
32. Власенко А.О. Застосування алгоритмів планування траєкторій у системах БПЛА // Системи управління та навігації. 2021. №3. С. 45-50.
33. Сидоров І.М. Алгоритми пошуку оптимального шляху для безпілотних систем // Авіаційно-космічна техніка і технологія. 2020. №6(169). С. 132-138.
34. Garcia J., Fernandez F. A Comprehensive Survey on Safe Reinforcement Learning. Journal of Machine Learning Research, 2015, 16: 1437-1480.
35. Zhang C., et al. Energy-Efficient Path Planning for UAV Swarms. IEEE Communications Magazine, 2019, 57(12): 132-137.
36. Gao Y., et al. 3D Path Planning for UAVs in Dynamic Environments. IEEE Access, 2020, 8: 61721-61733.
37. Kennedy J., Eberhart R. Particle Swarm Optimization. Proceedings of ICNN, 1995.
38. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. Pearson, 2020.
39. Sommerville I. Software Engineering. 10th ed. Pearson, 2015.
40. Lutz M. Learning Python. 5th ed. O’Reilly Media, 2013.

41. Goodrich M.T., Tamassia R. Algorithm Design and Applications. Wiley, 2014.