МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Олена НЕЧИПОРУК
«_____»_____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Програмний модуль управління польотом БПЛА з використанням машинного навчання

Виконавець: Забарющий Станіслав Григорович

Керівник: Кучеров Дмитро Павлович

Нормоконтролер: Тупота Євгеній Вікторович

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Факультет комп'ютерних наук та технологій
Кафедра інтелектуальних кібернетичних систем
Спеціальність 126 «Інформаційні системи та технології»
(шифр, найменування)

Освітньо–професійна програма «Інтелектуальні системи та технології»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олена НЕЧИПОРУК
«_____»_____2025 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Забарющого Станіслава Григоровича

(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема кваліфікаційної роботи «Програмний модуль управління польотом БПЛА з використанням машинного навчання»

затверджена наказом ректора від «28» серпня 2025 р. №1762/ст

2. Термін виконання роботи (проєкту): з 29.09.2025 до 31.12.2025

3. Вихідні дані до роботи (проєкту): мова програмування *C++*, середовище розробки *Visual Studio* 2022

4. Зміст пояснювальної записки:
1) Доцільність застосування методів машинного нання для управління польотом БПЛА
2) Методи машинного навчання та математичні моделі управління польотом БПЛА
3) Проектування програмного модуля управління
4) Тестування програмного модуля

5. Перелік обов'язкового графічного матеріалу:
1) Діаграма послідовності формування команд;
2) Організації даних відповідність карти та *Q*–тблиці;
3) Компонентна діаграма статичної структури модуля;
4) Алгоритм динамічного циклу навчання.

6. Календарний план–графік

№ пор.	Завдання	Термін виконання	Примітка
1	Ознайомитись з постановкою задачі дипломного проектування	01.11.2025 – 03.11.2025	
2	Вивчити спеціальну наукову літературу і документацію	03.11.2025 – 06.11.2025	
3	Проаналізувати методи штучного інтелекту і машинного навчання	06.11.2025 – 09.11.2025	
4	Написати розділ 1.	09.11.2025 – 19.11.2025	
5	Проаналізувати типові методи машинного навчання. Виконати проектування програмного модуля.	19.11.2025 – 23.11.2025	
6	Написати розділ 2.	23.11.2025 – 13.12.2025	
7	Розробити програмний модуль управління польотом БПЛА	13.12.2025 – 19.12.2025	
8	Написати розділи 3 і 4.	19.12.2025 – 21.12.2025	
9	Оформити пояснювальну записку	21.12.2025 – 23.12.2025	
10	Підготовити графічний демонстраційний матеріал	23.12.2025 – 25.12.2025	

7. Дата видачі завдання: “ 29 "вересня 2025 р

Керівник кваліфікаційної роботи _____ Кучеров Д.П.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Забарющий С.Г.

(підпис здобувача вищої освіти) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Модуль управління польотом БПЛА з використанням машинного навчання»: 90 с., 29 рис., 2 табл., 21 літературне джерело.

Об’єкт дослідження – автономне управління рухом безпілотного літального апарату у середовищі зі статичними перешкодами.

Предмет дослідження – навігація та планування траєкторії польоту на основі алгоритмів навчання з підкріпленням.

Мета кваліфікаційної роботи – удосконалення підходу до навігації БПЛА за умови відсутності сигналів супутникової навігації (*GPS*) шляхом організації навчання орієнтації в просторі та уникнення колізій на основі алгоритмів машинного навчання.

Методи проєктування – аналіз існуючих підходів, синтез алгоритмів управління на основі Марківських процесів прийняття рішень та алгоритму *Q–learning*, тестування розроблених підходів засобами об’єктно – орієнтованого та процедурного програмування мовою *C++ (C++17*) з використанням бібліотеки *STL*; прототипування в середовищі розробки Microsoft *Visual Studio*; моделювання топології місцевості.

До наукової новизни можна віднести такі рішення, які реалізовані в даному проєкті:

1. Удосконалення методу планування маршруту в дискретному середовищі шляхом адаптації табличного методу *Q–learning* з $\in$ –жадібною стратегією, що, на відміну від класичних алгоритмів пошуку, дозволяє агенту адаптуватися до топології карти без попереднього знання її структури.
2. Розробка алгоритму агрегації команд, який відрізняється від існуючих меншою кількістю команд для польотного контролера за рахунок векторизації дискретної послідовність вхідних даних.
3. Трансформація системи винагород для агента, у штрафи за кожен крок переміщення, що гарантує знаходження не тільки безпечний, а й енергетично вигідний маршрут для збереження заряду батареї БПЛА.

Практичне значення роботи полягає у створенні кросплатформеного програмного модуля, який не потребує застосування важких бібліотек машинного навчання і може бути інтегрований у діючі системи управління дронами для виконання місій у зонах з відсутнім сигналом *GPS*.

Ключові слова: БПЛА, МАШИННЕ НАВЧАННЯ, *Q–LEARNING*, АВТОНОМНА НАВІГАЦІЯ, *C++*

ЗМІСТ

ВСТУП

Актуальність теми. На сучасному етапі розвитку авіаційної техніки спостерігається стрімке зростання зацікавленості у використанні безпілотних літальних апаратів (БПЛА) як для військових, так і для цивільних цілей. Завдяки досягненням у галузі мікроелектроніки та обчислювальної техніки, сучасні дрони трансформувалися з простих дистанційно керованих механізмів у складні роботехнічні комплекси. Вони знаходять широке застосування у сферах аеророзвідки, моніторингу критичної інфраструктури, пошуково–рятувальних операціях та точному землеробстві.

Однак, зі збільшенням автономності БПЛА виникають нові виклики. Ключовою проблемою залишається залежність навігаційних систем від супутникового зв'язку (*GNSS/GPS*). В умовах щільної міської забудови, підземних комунікацій або застосування засобів радіоелектронної боротьби (РЕБ), традиційні методи позиціонування стають неефективними або повністю недоступними. У таких сценаріях критично важливою стає здатність бортового комп'ютера самостійно приймати рішення щодо маршруту руху, спираючись лише на дані про перешкоди.

Існуючі детерміновані алгоритми навігації (наприклад, A*, *Dijkstra*) вимагають значних обчислювальних ресурсів для перерахунку маршруту при зміні умов. З іншого боку, методи глибокого навчання забезпечують високу адаптивність, але потребують потужних графічних процесорів, що є критичним обмеженням для легких БПЛА.

У цьому контексті актуальність теми обумовлюється необхідністю розробки легковагових, але адаптивних програмного модуля управління на основі навчання з підкріпленням. Використання алгоритму *Q–learning* дозволяє створити систему, яка здатна самостійно навчатися оптимальній навігації в середовищі зі статичними перешкодами, забезпечуючи баланс між швидкодією та інтелектуальною гнучкістю. Створення такого модуля мовою *C++* без використання важких зовнішніх бібліотек

дозволяє імплементувати його на широкому спектрі вбудованих систем, що є важливим кроком до повної автономності БПЛА.

Мета –удосконалення підходу до навігації БПЛА за умови відсутності сигналів супутникової навігації (*GPS*) шляхом організації навчання орієнтації в просторі та уникнення колізій на основі алгоритмів машинного навчання.

Об'єкт дослідження –автономне управління рухом безпілотного літального апарату у середовищі зі статичними перешкодами.

Предмет дослідження – навігація та планування траєкторії польоту на основі алгоритмів навчання з підкріпленням.

Методи проектування –аналіз існуючих підходів, синтез алгоритмів управління на основі Марківських процесів прийняття рішень та алгоритму *Q–learning*, тестування розроблених підходів засобами об'єктно–орієнтованого та процедурного програмування мовою *C++ (C++17)* з використанням бібліотеки *STL*; прототипування в середовищі розробки Microsoft *Visual Studio*; моделювання топології місцевості.

Наукова новизна полягає у наступному:

Удосконалення методу планування маршруту в дискретному середовищі шляхом адаптації табличного методу *Q–learning* з $\in$ –жадібною стратегією, що, на відміну від класичних алгоритмів пошуку, дозволяє агенту адаптуватися до топології карти без попереднього знання її структури.

Розробка алгоритму агрегації команд, який відрізняється від існуючих меншою кількістю команд для польотного контролера за рахунок векторизації дискретної послідовність вхідних даних.

Трансформація системи винагород для агента у штрафи за кожен крок переміщення, що гарантує знаходження не тільки безпечний, а й енергетично вигідний маршрут для збереження заряду батареї БПЛА.

Практичне значення роботи полягає у створенні кросплатформеного програмного модуля, який не потребує застосування важких бібліотек машинного навчання і може бути інтегрований у діючі системи управління дронами для виконання місій у зонах з відсутнім сигналом *GPS*.

Особистий внесок здобувача.

Усі результати, викладені у дипломній роботі, отримані самостійно. Автором проведено аналіз літературних джерел, виконано математичну постановку задачі, написано програмний код модуля управління, розроблено тестові карти та проведено серію експериментів з налаштування алгоритму.

Апробація результатів дипломної роботи.

Основні положення та результати роботи доповідалися на:

– КАІ НАУ, Конференція ІТЛА, Київ, 2025 р.;

Публікації.

За темою дипломної роботи опубліковано 1 тезу доповідь:

– Міжнародна наукова–технічна конференція «Інтелектуальні технології лінгвістичного аналізу»: Тези доповідей. – К.: ДУ КАІ, 2025. – 180 с

Структура та обсяг роботи.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проведено доцільність застосування методів машинного навчання.

У другому розділі описано математичну модель середовища та алгоритм машинного навчання

У третьому розділі наведено деталі програмної реалізації та архітектуру модуля.

У четвертому розділі викладено результати тестування та експериментального дослідження параметрів навчання.

РОЗДІЛ 1
ДОЦІЛЬНИІСТЬ ЗАСТОСУВАННЯ МЕТОДІВ МАШИНОГО НАВЧАННЯ ДЛЯ УПРАВЛІННЯ ПОЛЬТОМ БПЛА

1.1. Сфера застосування систем управління польотом БПЛА

Безпілотні літальні апарати знаходять широке застосування у різних сферах, особливо там, де виконання завдань людиною є небезпечним або технічно складним. Це може бути аграрний моніторинг, патрулювання тяжко доступних територій, спостерігати за масовими подіями та усунення наслідків стихійних лих. Важливим фактором є те, що БПЛА можуть керуватися дистанційно за допомогою комп'ютера через радіозв'язок, що усуває необхідність фізичної присутності пілота. Але для цього повинен бути модуль для самостійного керування БПЛА.

Застосування модулів управління польотом з використанням машинного навчання відкриває нові можливості для підвищення ефективності роботи БПЛА. Адаптивні алгоритми навчання дозволяють системі оптимально контролювати рух апарата, обробляти великі обсяги даних у реальному часі та підлаштовувати керування під зміни умов польоту.

Створення таких систем управління охоплює визначення польотних завдань і параметрів руху, формування траєкторії та вибір алгоритмів, що забезпечують стійке й безпечне керування апаратом уздовж заданого маршруту. Оскільки умови польоту динамічні, управління БПЛА стає складнішою і більш витонченою задачею.

Принципи управління формують базові підходи та методи, що використовуються під час проектування систем керування безпілотними літальними апаратами.

Алгоритм повинен забезпечити компенсацію перехресних впливів та стабільність керування під час маневрування, включаючи складні просторові траєкторії з різкими змінами висоти та швидкості. Після аналізу ряду публікацій щодо застосування машинного навчання в управлінні БПЛА були визначені ключові перспективи їх впровадження (рис. 1.1). [3].

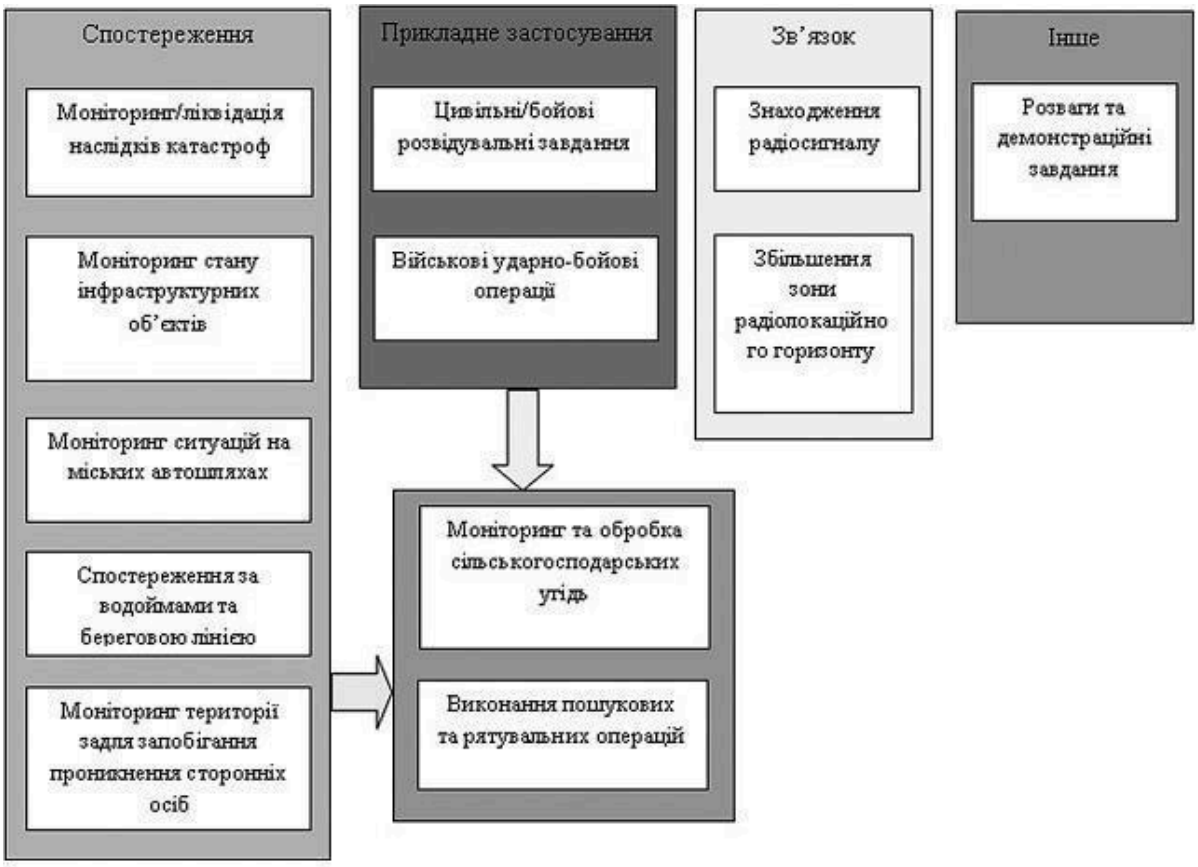


Рис. 1.1. Наявні та перспективні напрямки використання БПЛА з машинним навчанням

Подивимось на існуючі перспективні активності БПЛА з машинним навчанням напрями, їх можна розділити на такі основні функціональні групи: спостереження, прикладне застосування, зв'язок та ін.

1.1.1. Використання машинного навчання в БПЛА для моніторингу та ліквідації наслідків катастроф

Використання безпілотних літальних апаратів (БПЛА), оснащених алгоритмами машинного навчання, у сфері моніторингу та ліквідації наслідків

катастроф розглядається як один із найбільш перспективних напрямів їх практичного застосування. Особливо актуальним це є під час контролю ситуації на окремих територіях зі складними або небезпечними умовами експлуатації, де застосування автономних режимів польоту є більш ефективним, ніж безпосереднє керування оператором. Такий підхід дозволяє забезпечити безперервний аналіз даних та значно підвищити точність виявлення надзвичайних ситуацій [4].

Застосування автопілотованих БПЛА з елементами машинного навчання дає змогу охоплювати значно більші площі спостереження за коротший проміжок часу. Завдяки автоматизованій обробці інформації підвищується якість аналізу обстановки, що є критично важливим у разі виникнення природних або техногенних катастроф. Крім того, зменшується навантаження на операторів і підвищується загальна ефективність моніторингових систем.

В Розроблення подібних технологій розпочалося ще у 2000–х роках, а значний прогрес було досягнуто завдяки використанню літакоподібних БПЛА. Такі апарати продемонстрували високу результативність під час повітряного моніторингу водних ресурсів на великих відстанях, що підтвердило доцільність їх застосування для довготривалих та масштабних спостережень [4].

Регулярний моніторинг, що здійснюється за допомогою БПЛА, дозволяє своєчасно відстежувати паводкову ситуацію, визначати потенційні зони затоплення та виявляти порушення, зокрема незаконну діяльність у прибережних районах. Контроль охоплює не лише водні об’єкти, а й прилеглу берегову лінію. Дані, отримані з безпілотних апаратів, дають змогу оперативно оцінювати поточний стан території та прогнозувати подальший розвиток подій.

1.1.2. Використання машинного навчання в БПЛА для проведення пошуково рятувальних операцій

Застосування безпілотних літальних апаратів (БПЛА) для виконання операцій в екстремальних умовах і на важкодоступних територіях розглядається як один із найбільш перспективних напрямів розвитку безпілотних технологій з моменту формування самої концепції БПЛА. Станом на 2022 рік окремі безпілотні апарати вже залучаються до діяльності берегової охорони в низці країн, зокрема у США (рис. 1.2). У січні 2019 року було зафіксовано перший підтверджений випадок успішного порятунку людини із застосуванням безпілотного літального апарату [5].

Рис. 1.2. Рятувальний безпілотний конвертоплан *Eagle Eye*, розроблений фірмою *Bell* для берегової охорони США

Згідно з даними служб, що використовують БПЛА під час проведення рятувальних операцій, середній час реагування на надзвичайні ситуації скорочується з 5–10 хвилин до приблизно 1,5 хвилини. Зокрема, перший задокументований випадок порятунку людини поблизу узбережжя Австралії показав, що проміжок часу між передачею команди БПЛА та досягненням визначеної цілі становив лише 70 секунд, що з урахуванням дистанції та складних погодних умов було оцінено як надзвичайно високий результат.

Використання БПЛА, оснащених алгоритмами машинного навчання, для розв’язання подібних завдань є ще більш перспективним, оскільки це забезпечує

розширену функціональність систем і, зокрема, можливість виконання рятувальних операцій без безпосередньої або з мінімальною участю оператора. Такий підхід дозволяє оперативніше реагувати на надзвичайні події та підвищує ефективність пошукових робіт, зокрема у випадках авіаційних катастроф у важкодоступних районах падіння. На сьогодні повноцінні працездатні системи, які б дозволяли масове застосування БПЛА з машинним навчанням у рятувальних операціях, ще не впроваджені, однак активні дослідження та розробки в цьому напрямі здійснюються виробниками безпілотних апаратів з Китаю, Ізраїлю, США, Канади та інших країн.

1.1.3. Використання БПЛА з машинним навчанням для пошуку радіосигналу

У даному випадку основним завданням застосування безпілотних літальних апаратів (БПЛА) є оперативне визначення місця розташування джерела радіосигналу, навіть за умов складної або важкодоступної місцевості. Наразі подібні технічні рішення вже перебувають на стадії експериментальних випробувань у низки виробників безпілотних систем.

Використання таких систем на базі БПЛА з алгоритмами машинного навчання дає змогу підвищити точність локалізації сигналу та суттєво скоротити час, необхідний для його виявлення. Подібні технології можуть застосовуватися для пошуку радіосигналів бортових самописців повітряних суден, сигналів морських об’єктів у аварійних ситуаціях, а також для виявлення осіб, які зникли у віддалених або важкодоступних районах, чи втратили зв’язок під час надзвичайних подій. Застосування таких систем значно підвищує ефективність проведення пошуково–рятувальних операцій як на суші, так і в морському середовищі, а також відкриває можливості їх використання у низці військових завдань.

Крім того, ефективна експлуатація БПЛА в екстремальних умовах і на важкодоступних територіях розглядається як один із найбільш перспективних напрямів розвитку безпілотних технологій. Станом на сьогодні існує успішний практичний досвід застосування БПЛА у підрозділах берегової охорони ряду держав,

зокрема США, що підтверджується зафіксованими випадками порятунку людей за допомогою безпілотних літальних апаратів.

Зазначається, що залучення БПЛА дозволяє суттєво скоротити час реагування на надзвичайні ситуації, особливо під час рятувальних операцій. Так, у порівнянні з традиційними методами роботи аварійно–рятувальних служб, використання груп БПЛА може зменшити час реагування з 5–10 хвилин до приблизно 1,5 хвилини, що робить їх надзвичайно ефективним інструментом у ліквідації наслідків катастроф і надзвичайних ситуацій.

1.1.4. Використання БПЛА з машинним навчанням для виконання цивільних або бойових розвідувальних завдань

Застосування безпілотних літальних апаратів для виконання цивільних і бойових розвідувальних завдань є одним із основних напрямів їх використання. У межах таких завдань БПЛА демонструють високу ефективність, оскільки здатні забезпечувати оперативний і масштабний збір розвідувальної інформації.

У цивільній сфері, зокрема під час моніторингу дорожнього руху, патрулювання лісових територій або здійснення аграрного контролю, БПЛА можуть узгоджувати свої дії для більш результативного виконання поставлених завдань. Це дає змогу швидко отримувати актуальні дані, сприяє оптимальному використанню ресурсів і підвищенню загальної ефективності відповідних процесів.

Під час проведення бойових розвідувальних операцій безпілотники застосовуються для збору значних обсягів інформації на великих площах. Координуючи свої переміщення та функції, такі апарати забезпечують формування цілісного й об’єктивного уявлення про поточну бойову обстановку. Крім того, БПЛА відіграють важливу роль у сучасних бойових діях, оскільки здатні виконувати розвідувальні завдання в районах, недоступних або небезпечних для традиційних засобів спостереження.

США наразі інвестує в безпілотник на основі літака для вирішення бойових задач. Головною ідеєю створення *XQ–67A* є підтримка винищувачів, завдяки здатності як до дистанційного пілотування, так і до автономного польоту.

Рис. 1.3. Безпілотний літальний апарат *XQ–67A*

1.1.5. Використання БПЛА з машинним навчанням для виконання військових ударно бойових завдань

Застосування безпілотних літальних апаратів, у яких використовуються методи машинного навчання, є одним із ключових напрямів сучасного військово–технічного розвитку. У цьому контексті використання груп БПЛА забезпечує низку стратегічних переваг і розширює можливості виконання різноманітних завдань у зоні бойових дій.

Однією з головних переваг залучення БПЛА у військових операціях є їх здатність до скоординованої взаємодії, що дозволяє ефективно розв’язувати як стратегічні, так і тактичні завдання. Робота безпілотників забезпечує високий рівень маневреності та адаптивності під час бойових дій. БПЛА можуть застосовуватися для нанесення точкових і узгоджених ударів по об’єктах противника, що сприяє підвищенню точності ураження та зменшенню побічних втрат. Крім того, такі апарати широко використовуються для проведення розвідувальних операцій, забезпечуючи оперативне отримання інформації про поточну обстановку на полі бою.

Використання БПЛА дозволяє суттєво підвищити інтенсивність і ефективність військових операцій, одночасно знижуючи ризики для особового складу та створюючи додаткові можливості для стратегічного планування і реалізації бойових завдань у сучасних умовах ведення війни.

Найбільш значні досягнення у сфері розвитку та застосування БПЛА пов'язані саме з військовою галуззю. На сьогодні серед країн, що займають провідні позиції у впровадженні безпілотних технологій для потреб збройних сил, особливо вирізняються США та Ізраїль. Як приклад можна навести проєкт *DARPA Gremlins*, у межах якого розроблені апарати *Gremlin*, здатні розвивати швидкість у межах 864–987 *км/год*. Тривалість польоту таких безпілотників становить від однієї до трьох годин, а радіус дії – 555–926 *км.* Основними завданнями апаратів *Gremlin* є виконання розвідувальних місій та здійснення радіоелектронного придушення цілей. Вони можуть підлітати до об'єктів супротивника якомога ближче і при масовому застосуванні «забивати» канали проти–повітряної оборони [6].

Low–Cost UAV Swarming Technology (*LOCUST*). Що стосується програми застосування над дешевих безпілотних літальних апаратів Науково–дослідної лабораторії Військово–морських Сил США «*LOCUST*», то тут планується створити кординований стрій, в який увійде до 30 БПЛА. У березні 2017 р. розробники «*LOCUST*» створили дослідний «рій» з 9 апаратів *Coyote* (рис. 1.4) і систему їх запуску з борту кораблів.

Рис. 1.4. БПЛА *Coyote* –безпілотний літальний апарат «*LOCUST*»

Порівняльна характеристика апаратів, застосованих для виконання військових операцій спостереження та ураження цілей в складі групи, представлена в табл. 1.1

Система «*LOCUST*» являє собою пусковий комплекс, що складається з набору трубчастих контейнерів, у кожному з яких розміщується окремий літальний апарат. Після старту з пускової установки дрон у повітрі автоматично розкриває крила та інші аеродинамічні елементи. Довжина апарата становить близько 91 см, маса — трохи менше 6 кг; він оснащений електричним двигуном і здатний виконувати польотні завдання на дальності до 37 км при максимальній висоті польоту до 7600 м. Після запуску безпілотники, випущені з однієї пускової установки, можуть координувати свої дії між собою для реалізації різних військових сценаріїв, зокрема завдань наступального та оборонного характеру.

Військовими фахівцями було проведено серію експериментальних випробувань з метою оцінювання реакції систем протиповітряної оборони на раптову атаку групи з 5–10 безпілотних літальних апаратів, які здійснювали напад на військовий корабель з різних напрямків. Через малі габарити БПЛА радіолокаційні засоби виявляли їх наближення лише на надзвичайно коротких дистанціях –менше ніж два кілометри. За умови швидкості польоту безпілотників близько 250 *км/год* максимальний час від моменту виявлення до ураження цілі не перевищував 15 секунд. При цьому жодні засоби радіоелектронного пригнічення не змогли порушити функціонування системи керування БПЛА.

1.1.6. Використання груп БПЛА для розширення зони радіолокаційної видимості

Одним із ключових завдань сучасних систем керування та моніторингу є усунення обмежень, пов’язаних із радіолокаційним горизонтом, величина якого зумовлюється кривизною земної поверхні та особливостями рельєфу місцевості. Застосування безпілотних літальних апаратів дозволяє суттєво розширити зону огляду, оскільки вони можуть виступати мобільними платформами для розміщення радіолокаційних та сенсорних систем.

Використання методів машинного навчання у цій сфері дає можливість забезпечити оптимальне управління польотом з урахуванням рельєфу, погодних умов

та наявності перешкод. Завдяки інтелектуальним алгоритмам БПЛА здатні самостійно підбирати оптимальні траєкторії та висоту польоту, що дозволяє максимально збільшити зону спостереження при мінімальних витратах енергії.

Крім того, машинне навчання підвищує точність обробки інформації з бортових сенсорів і радіолокаційних систем. Це зменшує рівень шумів і помилкових сигналів, покращує точність визначення об'єктів та формує більш чітку картину навколишнього середовища. Таким чином, інтеграція БПЛА та алгоритмів машинного навчання створює нові можливості для побудови гнучких і масштабованих систем, здатних значно збільшувати радіолокаційний горизонт та підвищувати якість управління повітряним простором.

1.1.7. Використання машинного навчання в управлінні польотом БПЛА для демонстраційних та розважальних цілей

Розважальна сфера стала одним із перших напрямів, де було продемонстровано ефективність використання машинного навчання для управління польотом безпілотних літальних апаратів. Починаючи з невеликих груп апаратів у 2012 році, сучасні технології машинного навчання дозволяють координувати рух сотень і навіть тисяч дорнів у повітрі. Завдяки алгоритмам навчання дорни здатні синхронно виконувати зліт, посадку, зміну формацій та інші маневри без попереднього жорсткого програмування.

Методи машинного навчання забезпечують:

1) адаптивне формування траєкторій руху дорнів;

2) автоматичне уникнення зіткнень;

3) корекцію польоту в умовах зовнішніх перешкод (вітер, зміна висоти);

4) синхронізацію виконання колективних фігур і світлових ефектів.

Застосування таких алгоритмів у розважальних подіях дало можливість створювати масштабні світлові шоу, де БПЛА утворюють динамічні зображення в небі. Використання машинного навчання дозволяє апаратам швидко пристосовуватися до зміни сценарію та підтримувати узгодженість руху навіть за наявності випадкових відхилень.

Таким чином, демонстраційні проекти показують, що машинне навчання може ефективно застосовуватися для керування польотом великої кількості БПЛА одночасно. Отримані результати мають перспективу використання не лише у сфері шоу та розваг, а й в транспортній логістиці, моніторингу та інших практичних галузях.

1.1.8. Використання машинного навчання для управління БПЛА при моніторингу ситуацій на міських автошляхах

За умов високої щільності сучасних мегаполісів, де кількість автотранспорту є значною, а дорожня обстановка характеризується динамічністю та непередбачуваними змінами, класичні системи моніторингу й оповіщення не завжди здатні забезпечити необхідний рівень швидкодії та достовірності. У зв’язку з цим перспективним напрямом є застосування безпілотних літальних апаратів, споряджених системами керування польотом, що базуються на методах машинного навчання. Такі алгоритми дозволяють адаптивно змінювати траєкторії руху дрону залежно від поточної дорожньої обстановки та прогнозувати розвиток ситуацій у режимі реального часу.

Завдяки застосуванню машинного навчання БПЛА можуть не лише фіксувати місце виникнення інцидентів, але й передбачати ймовірні транспортні затори, аварії або інші критичні події. Це дає можливість екстреним службам, громадському транспорту та таксі отримувати оперативні дані з високою точністю. Крім того, адаптивне управління польотом дозволяє охопити більшу територію міста, мінімізуючи дублювання маршрутів між дронами та забезпечуючи комплексний моніторинг з різних точок спостереження.

У результаті використання технологій машинного навчання в управлінні польотом БПЛА значно підвищується ефект міських систем моніторингу, зменшується час реагування на надзвичайні ситуації та покращується координація між усіма учасниками дорожнього руху. Станом на 2022 р. у більш ніж 50 містах світу активно тестуються та впроваджуються подібні інтелектуальні системи управління польотом дронів.

Основні сфери використання БПЛА в містах:

1. Моніторинг дорожнього руху: БПЛА можуть використовуватися для відстеження заторів, аварій, а також для контролю за дотриманням правил дорожнього руху;
2. Спостереження за громадською безпекою: БПЛА можуть використовуватися для моніторингу публічних заходів, а також для запобігання злочинам і тероризму;
3. Інспектування інфраструктури: БПЛА можуть використовуватися для інспектування мостів, доріг, будівель та інших об'єктів інфраструктури.
4. Картографування та геодезія: БПЛА можуть використовуватися для створення 3D–карт і моделей місцевості;
5. Доставка: БПЛА можуть використовуватися для доставки невеликих вантажів, наприклад, ліків або продуктів харчування.

Переваги використання БПЛА:

1. Ефективність: БПЛА можуть виконувати завдання швидше і дешевше, ніж люди;
2. Безпека: БПЛА можуть використовуватися в небезпечних або важкодоступних місцях;
3. Точність: БПЛА можуть збирати дані з високою точністю;
4. Гнучкість: БПЛА можуть бути оснащені різними датчиками та камерами для виконання

Виклики використання БПЛА:

1. Конфіденційність: БПЛА можуть збирати особисті дані людей, що може призвести до порушення конфіденційності.
2. Безпека: БПЛА можуть бути збиті або захоплені, що може призвести до ризиків для людей і майна.
3. Регулювання: Існують різні правила та обмеження на використання БПЛА, які можуть ускладнити їх експлуатацію.

Під час розгортання подібних систем спостереження в міському середовищі доцільним є доповнення їх модулями автоматичного розпізнавання осіб і

транспортних засобів. У процесі використання в підрозділах поліції та інших служб безпеки такі рішення можуть ефективно застосовуватися для розшуку людей, контролю за правопорушниками, а також для виявлення й відстеження автомобілів.

1.1.9. Використання машинного навчання для управління польотом БПЛА в аграрній сфері

Безпілотні авіаційні комплекси, у яких інтегровані методи машинного навчання, набувають все більшого значення як інструмент підвищення результативності аграрного виробництва. До складу таких комплексів можуть входити безпілотні літальні апарати (БПЛА), вимірювальні датчики, засоби побудови карт місцевості, а також програмні модулі, призначені для обробки й аналізу великих обсягів даних. Завдяки використанню алгоритмів машинного навчання управління польотом БПЛА стає адаптивним, що дозволяє автоматизувати аграрні процеси та зменшити вплив людського фактору. Ось кілька напрямів застосування таких систем [7]:

1. Моніторинг стану полів. Застосування алгоритмів машинного навчання до знімків і сенсорних даних, отриманих із БПЛА, дозволяє точно визначати вологість ґрунту, рівень живлення культур, а також прогнозувати врожайність;
2. Розпилення добрив та пестицидів. Завдяки аналізу в режимі реального часу дрони можуть регулювати траєкторії польоту та висоту розпилення, забезпечуючи точність і рівномірність покриття, що зменшує витрати ресурсів і шкоду довкіллю;
3. Картографування та аналіз даних. Машинне навчання дозволяє створювати детальні карти продуктивності полів, ідентифікувати ділянки зі зниженим потенціалом та прогнозувати їх зміни з урахуванням кліматичних і ґрунтових факторів;
4. Моніторинг росту та розвитку рослин. Алгоритми розпізнавання образів дозволяють своєчасно виявляти ознаки хвороби, шкідників або стресу рослин, що забезпечує швидке реагування та зменшує ризики втрати врожаю.

5. Початкове зорове обстеження. БПЛА з ML–моделями можуть автоматично класифікувати пошкодження посівів після засухи, повені чи інших природних катастроф;
6. Оптимізація поливу. Алгоритми машинного навчання аналізують потребу у волозі на різних ділянках та керують польотом дронів для точкової подачі води, забезпечуючи оптимальні умови для розвитку культур.

Станом на початок 2021 року, відповідно до відкритих офіційних джерел, основним напрямом застосування безпілотних літальних апаратів в Україні залишається аграрна галузь. Орієнтовно близько однієї чверті від загальної кількості БПЛА, що використовуються в державі, залучені до потреб сільського господарства. Така тенденція обумовлена функціональною універсальністю безпілотних систем, які дозволяють виконувати широкий спектр завдань — від дистанційного збору інформації та контролю стану угідь до автоматизованого внесення добрив і здійснення зрошення (рис. 1.5).

Рис. 1.5. Використання БПЛА у сільському господарстві

У процесі виконання таких завдань безпілотна авіація з технологіями машинного навчання поступово заміщає спеціалізовані пілотовані апарати [8]. Подальший розвиток може передбачати використання груп БПЛА, що працюють на

основі навчальних моделей для координації своїх маршрутів і розподілу завдань. Це дозволить збільшити охоплення території, підвищити точність збору даних і забезпечити створення єдиної комплексної картини стану сільськогосподарських угідь. Крім того, алгоритми машинного навчання роблять можливим прогнозування надзвичайних ситуацій, таких як пожежі, підтоплення, поширення шкідників чи інші загрози, забезпечуючи швидке реагування та мінімізацію втрат.

1.2. Проблеми та виклики управління польотом в реальних умовах

Управління польотом із застосуванням методів машинного навчання (ML) відкриває великі перспективи для автономних і на пів автономних літальних апаратів. Водночас, у реальних умовах виникає багато складнощів, які потрібно враховувати при розробці систем управління польотом. Нижче наведено основні проблеми та виклики, з ілюстраціями з сучасних досліджень.

Основні проблеми та виклики:

1) Нелінійність динаміки та невизначеність середовища.

   Динаміка БПЛА має нелінійний характер і суттєво залежить від зовнішніх факторів (вітер, турбулентність, зміни навантаження). Традиційні *PID*–контролери часто не забезпечують необхідної точності;

   Сучасні *ML* підходи, наприклад *Neural–Fly*, показують здатність адаптуватись до різних вітрових умов і перевершують класичні методи.

2) Обмеження даних і їх різноманітність;

   Для навчання моделей потрібні великі та різноманітні набори даних. Проблемою є «розрив між симуляцією та реальністю».

   Приклад –робота [8], де підкреслюється важливість врахування реальних затримок та шумів при навчанні;

3) Обмежені обчислювальні ресурси і енергія.

   Глибокі *ML*–моделі вимагають значних обчислювальних ресурсів і енергії, що обмежує їх застосування на борту БПЛА. У роботі підкреслюється

необхідність досягнення балансу між складністю алгоритмічних рішень і обчислювальними можливостями апаратного забезпечення.

4) Стійкість до несподіваних подій і наявність помилок

У реальних умовах можливих збоїв датчиків або актуаторів, а також зовнішні збурення. *RL* –методи продемонстрували здатність підтримувати працездатність навіть при несправностях.

1.3. Сучасні підходи до розв'язання проблеми керування

Сучасні методи управління польотом безпілотних літальних апаратів із застосуванням машинного навчання спрямовані на підвищення автономності, точності та стійкості БПЛА в реальних умовах. Основними підходами є використання глибокого навчання (*Deep Learning*) та методів підкріплювального навчання (*Reinforcement Learning, RL*), які дозволяють системам самостійно адаптуватися до змінних умов середовища та динаміки апарату.

Одним із ефективних сучасних підходів є гібридне управління, що поєднує класичні алгоритми регулювання (*PID, LQR*) із моделями машинного навчання. Такі гібридні системи здатні забезпечувати стабільність польоту за стандартних умов і водночас адаптуватися до несподіваних змін, наприклад, поривів вітру, зміни навантаження чи часткових несправностей датчиків [9].

Іншим напрямом є використання моделей, навчених у симуляторі, з подальшим перенесенням у реальні умови (*sim–to–real transfer*). Це дозволяє зменшити ризик пошкодження апарату під час експериментів і забезпечити швидку адаптацію до нових умов без додаткового тривалого збору даних у реальному світі [9].

Таким чином, сучасні підходи до управління польотом із застосуванням машинного навчання поєднати класичні методи регулювання з алгоритмами глибокого навчання та підкріплювального навчання, що дозволяє досягти високої точності, стійкості та адаптивності системи.

1.4. Особливості використання штучного інтелекту та машинного навчання

Використання штучного інтелекту (ШІ) та машинного навчання (ML) у системах управління польотом безпілотних літальних апаратів дозволяє підвищити автономність, адаптивність та точність польотів у складних і непередбачуваних умовах. Основна перевага таких систем полягає у здатності навчатися на основі великих масивів даних і самостійно коригувати поведінку апарату під час польоту без прямого втручання оператора.

Особливості застосування *ML* у керуванні польотом включають:

1. Адаптивність до зовнішніх умов. Моделі, навчені на історичних даних або у симуляторах, можуть швидко адаптуватися до змін вітру, навантаження чи несправностей датчиків.
2. Прогнозування та компенсація помилок. ШІ дозволяє передбачати можливі відхилення траєкторії та коригувати управління у реальному часі, що підвищує стабільність польоту.
3. Скорочення потреби у ручному налаштуванні. Системи на основі *ML* здатні самостійно оптимізувати параметри контролера, зменшуючи залежність від людських факторів та підвищуючи продуктивність розробки.
4. Інтеграція з класичними методами управління. Часто використовується гібридний підхід, коли ML –моделі доповнюють класичні алгоритми *PID* або LQR, забезпечуючи баланс між стабільністю та адаптивністю.
5. Таким чином, використання ШІ та *ML* у системах управління польотом відкриває нові можливості для автономного керування БПЛА, підвищення безпеки та ефективності польотів.

1.5. Критерії ефективності та якості управління

При оцінці ефективності систем управління польотом БПЛА з використанням машинного навчання основними критеріями є точність слідування траєкторії,

стабільність польоту, адаптивність до змін середовища та енергетична точність.

1. Точність слідування траєкторії. Моделі *ML* повинні забезпечувати мінімальні відхилення від заданої траєкторії, навіть у складних умовах (вітер, зміни навантаження, несправності датчиків). Для оцінки використовують середньоквадратичне відхилення (*RMSE*) або середню абсолютну помилку (*MAE*) між запланованою та реальною траєкторією [10].
2. Стабільність і плавність управління. Контролер повинен уникати різких стрибкоподібних команд, які можуть призвести до коливань чи пошкоджень апарату. Часто використовується показник *variance control signals*, який вимірює розкид команд управління.
3. Адаптивність та стійкість до збурень. Система повинна ефективно реагувати на несподівані зміни середовища та часткові несправності. Критерієм якості є здатність зберегти працездатність і точність польоту при виникненні збурень.
4. Енергетична ефективність. Важливий критерій для літальних апаратів з обмеженою ємністю батареї –управління повинно забезпечувати оптимальне використання енергії без надмірного навантаження на актуатори.

Таким чином, оцінка ефективності систем управління польотом із *ML* включає кількісні показники точності, стабільності та адаптивності, що дозволяє порівнювати різні алгоритми та оберти оптимальні для практичного застосування.

1.6. Висновки до розділу

У результаті аналізу проблем та сучасних підходів до управління польотом безпілотних літальних апаратів встановлено, що традиційні методи управління мають обмежену точність у реальних умовах через складність середовища, наявність зовнішніх збурень та обмеженість обчислювальних ресурсів. Це обумовлює потребу у використанні новітніх технологій, зокрема штучного інтелекту та машинного навчання.

Сучасні дослідження доводять, що алгоритми машинного навчання здатні забезпечити адаптивність, стійкість та високу точність управління польотом навіть у складних і динамічних умовах. Використання методів глибинного навчання та підкріплювального навчання дозволяє зменшити відхилення від заданої траєкторії, покращити точність та підвищити рівень автономності БПЛА.

Основними критеріями якості управління є точність відтворення траєкторії, стабільність роботи системи, адаптивність до змін середовища та ефективне використання енергоресурсів. Саме за цими показниками можна об’єктивно оцінювати ефективність різних алгоритмів та систем управління.

Таким чином, застосування методів машинного навчання у сфері управління польотом БПЛА відкриває нові можливості для створення більш надійних, гнучких та ефективних систем, що є важливим кроком у розвитку інтелектуальних автономних технологій.

РОЗДІЛ 2
МЕТОДИ МАШИННОГО НАВЧАННЯ ТА МАТЕМАТИЧНІ МОДЕЛІ УПРАВЛІННЯ ПОЛЬТОМ БПЛА

2.1. Огляд методів машинного навчання в задачах навігації БПЛА

Сучасні безпілотні літальні апарати (БПЛА) все частіше переходять від дистанційного керування оператором до повної або часткової автономності. Ключовою проблемою при цьому залишається навігація – здатність апарату самостійного визначати своє положення, будувати маршрут та уникати перешкод у динамічному середовищі. Традиційні алгоритмічні методи (такі як A* або PID–регулювання) ефективні у стабільних умовах, проте вони часто не здатні адаптуватися до змінних факторів середовища. [11] Як зазначають у своєму огляді Карріо та ін. [16], саме інтеграція алгоритмів машинного навчання дозволяє дронам виконувати складні місії без *GPS*–навігації, покладаючись лише на бортові сенсори та обчислювальні модулі.

Саме тому методи штучного інтелекту (ШІ) та машинного навчання (*Machine Learning, ML*) набувають критичного значення в задачах управління польотом.

Методи машинного навчання в контексті БПЛА можна класифікувати на три основні категорії: навчання з учителем (*Supervised Learning*), навчання без учителя (*Unsupervised Learning*) та навчання з підкріпленням (*Reinforcement Learning*). Кожен із цих підходів має свою специфіку застосування в навігаційних модулях.

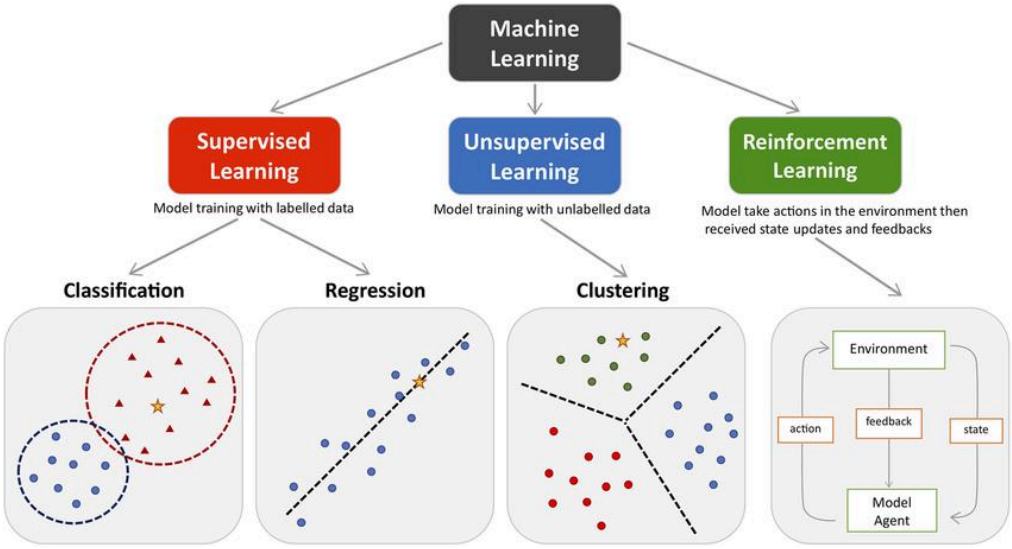


Рис. 2.1. Класифікація методів машинного навчання

2.1.1. Навчання з учителем: Нейронні мережі

Навчання з учителем передбачає наявність розміченого набору даних, на якому алгоритм "вчиться" приймати рішення. У контексті БПЛА найбільш поширеним інструментом тут є глибокі нейронні мережі (*Deep Neural Networks, DNN*) та згорткові нейронні мережі (*CNN*).

Основна сфера Основна сфера застосування *CNN* –це комп'ютерний зір (*Computer Vision*). Якщо БПЛА оснащений камерою, нейромережа може в реальному часі класифікувати об'єкти (дерева, будівлі, інші дрони) та передавати цю інформацію в модуль управління.

Однак, використання глибоких нейронних мереж для безпосереднього управління польотом має суттєві обмеження:

1. Обчислювальна складність: Обробка відеопотоку та робота багатошарових мереж вимагає потужних графічних процесорів (*GPU*), що складно реалізувати на борту легких дронів через обмеження ваги та енергоспоживання;
2. Залежність від даних: Для навчання такої системи потрібні тисячі годин польотних даних, розмічених експертами;

3. Ефект «чорної скриньки»: У системах критичної важливості, до яких відносяться БПЛА, важко інтерпретувати, чому саме нейромережа прийняла те чи інше рішення, що ускладнює налагодження (*debug*) програмного коду.

2.1.2. Алгоритм на основі древ рішень та випадкових лісів

Дерева рішень та ансамблеві методи, такі як випадковий ліс (*Random Forest*), використовуються для задач класифікації станів системи або регресійного аналізу. В управлінні польотом ці методи часто застосовуються для:

1. Діагностики несправностей (наприклад, визначення відмови двигуна за даними вібрації);
2. Вибору режиму польоту (наприклад, перемикання між режимом економії енергії та режимом маневрування залежно від вітру).

Перевагою цих методів є висока швидкість роботи та інтерпретованість результатів (прослідковується логіка "якщо–то"). Проте для задачі побудови маршруту в невідомому середовищі вони підходять менше, оскільки простір станів при навігації є занадто великим та неперервним, що призводить до надмірного розростання структури дерева і зниження ефективності.

2.1.3. Навчання з підкріпленням

Третій, і найбільш перспективний для задач автономної навігації підхід –це навчання з підкріпленням (*Reinforcement Learning, RL*). На відміну від навчання з учителем, *RL* не потребує заздалегідь підготовлених прикладів правильної поведінки. [10] Дослідження Кобера та Пітерса підтверджують, що *RL* є найбільш природним підходом для робототехніки, оскільки він дозволяє агенту адаптуватися до фізики польоту та непередбачуваних перешкод через метод проб і помилок [17].

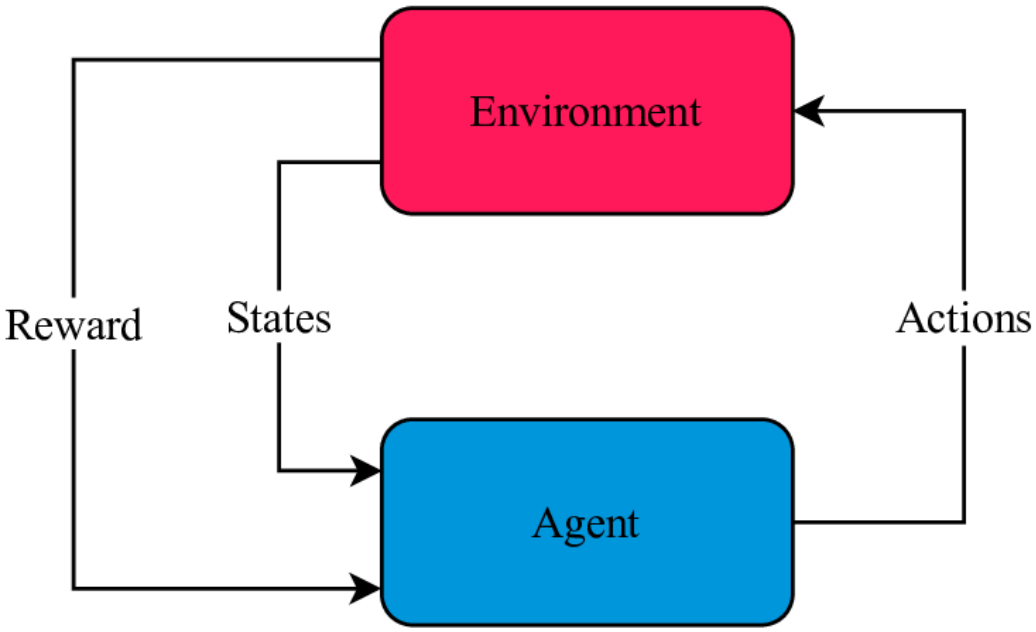


Рис. 2.2. Схема взаємодії агента та середовища в навчанні з підкріпленням

Суть методу полягає у взаємодії двох сутностей: Агента (БПЛА) та Середовища (фізичний простір або його симуляція, карта перешкод). Процес навчання відбувається циклічно:

1. Агент оцінює поточний стан середовища (S_t);
2. На основі своєї стратегії (політики) агент виконує дію (A_t) –наприклад, зміщується на одну клітинку вперед;
3. Середовище реагує на дію, переходячи в новий стан $\left(S_{\{t+1\}}\right)$ та повертаючи агенту числову винагороду або штраф (R_t).

Метою агента є максимізація сумарної винагороди за весь час роботи. Це ідеально відповідає задачі навігації дрона, де:

1. Досягнення цільової точки –велика позитивна нагорода;
2. Зіткнення з перешкодою –великий штраф;
3. Витрата часу/енергії на кожен крок –невеликий штраф.

Одним із класичних та ефективних алгоритмів *RL* є *Q–learning*. Це метод без модельного навчання (*model–free*), що дозволяє агенту вивчити оптимальну стратегію поведінки, просто досліджуючи простір.

Переваги *RL* та *Q–learning* для дипломного проекту:

1. Адаптивність: Агент може знаходити нестандартні шляхи обходу перешкод, які важко запрограмувати жорсткою логікою "*if–else*".
2. Обчислювальна точність: Табличний метод *Q–learning* (який реалізовано в даній роботі на мові *C++*) не вимагає складних матричних обчислень, характерних для нейромереж, і може працювати на вбудованих системах (мікроконтролерах) у реальному часі.
3. Робота з дискретним простором: Навігаційні карти часто представляються у вигляді сітки (*Grid Map*), що прямо корелює з математичною моделлю *Q–learning,* де простір станів є дискретним набором клітинок.

2.1.4. Підсумки до підрозділу

Аналіз методів показує, що хоча нейронні мережі є потужним інструментом для обробки сенсорної інформації, для задачі прийняття рішень щодо маршруту в умовах обмежених обчислювальних ресурсів найбільш доцільним є використання навчання з підкріпленням.[11] Зокрема, алгоритм *Q–learning* дозволяє реалізувати автономний пошук шляху, базуючись на системі винагород, що відповідає меті даної роботи – створенню ефективного модуля управління польотом.

В дипломному проекті буде використано саме підхід *Q–learning*, реалізований програмно без використання важких бібліотек машинного навчання, що забезпечить високу швидкодію та переносність коду.

2.2. Метод *Q–learning*: математична модель та алгоритм

Метод *Q–learning* (*Q*–навчання) є одним із найбільш ефективних та широко застосовуваних алгоритмів у класі методів навчання з підкріпленням. Вперше запропонований К. Уоткінсом[12]. Він належить до групи *off–policy* алгоритмів, що означає здатність агента навчатися на основі дій, які не обов'язково відповідають поточній стратегії, а також до групи *model–free* методів, оскільки агент не потребує

попереднього знання моделі середовища (правил фізики, карти перешкод тощо), а вивчає її безпосередньо через взаємодію.

Дослідження Кобера та Пітерса вказують, що RL є найбільш природним підходом для робототехніки, оскільки він дозволяє агенту адаптуватися до фізики польоту та непередбачуваних перешкод через метод проб і помилок [18]. У дипломномній роботі *Q–learning* обрано як базовий алгоритм для модуля управління польотом БПЛА завдяки його здатності знаходити глобально оптимальні маршрути в дискретних середовищах з мінімальними обчислювальними витратами.

2.2.1. Формалізація середовища як Марковського процесу прийняття рішень

Математичною основою *Q–learning* є теорія Марковських процесів прийняття рішень. Задачу навігації БПЛА можна описати кортежем (S, A, P, R, γ), де:

1. $S(States)$ –скінченна множина станів середовища. У розробленому програмному модулі простір станів представлено двовимірною сіткою (картою), де кожен стан $s \setminus S$ відповідає координатам дрона (x, y).
2. A $(Actions)$ –скінченна множина можливих дій агента. У реалізації визначено 4 дискретні дії:

$$A = \{Forward(0), Right(1), Back(2), Left(3)\}$$

3. R $(Reward\ Function)$ –функція винагороди $R: S \times A \rightarrow R$. Вона визначає числовий відгук середовища на дію агента.
4. γ $(Discount\ Factor)$ –коефіцієнт дисконтування $(0 \leq \gamma \leq 1)$, який визначає важливість майбутніх винагород порівняно з миттєвими.

2.2.2. *Q–функція* та рівняння Беллмана

Ключовим поняттям алгоритму є *Q–функція* або функція цінності дії. Значення $Q(s,a)$ визначає очікувану сумарну дисконтовану винагороду, яку отримає агент, якщо, перебуваючи у стані s, він виконає дію a, і надалі буде діяти оптимально. [13]

Теоретичне обґрунтування збіжності цієї функції до оптимальних значень детально розглянуто у праці Кліфтона та Лабера, де доведено, що при нескінченній кількості відвідувань кожного стану Q–значення гарантовано сходяться до Q^*[19].

Математично це виражається так:

$$Q(s,a) = E[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a]$$

У програмній реалізації Q–функція представлена у вигляді таблиці (*Look–up Table*) або тривимірного масиву *Q*[*height*][*width*][*actions*], де комірки зберігають дійсні числа, що постійно оновлюються в процесі навчання.

Процес оновлення значень у таблиці базується на рівнянні Беллмана. Ітераційна формула для перерахунку Q–значень має вигляд:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r_{t+1} + \gamma \cdot max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$$

де: $Q(s_t, a_t)$ – поточне значення цінності для стану s_t та дії a_t;

α (*Learning Rate*) – швидкість навчання $(0 < a < 1)$. Визначає, наскільки нові дані замінюють старі знання. У розробленому модулі прийнято $\alpha = 0.1$;

r_{t+1} – миттєва винагорода, отримана після виконання дії;

$\max_{a'} Q(s_{t+1}, a')$ – оцінка максимальної можливої винагороди у наступному стані

s_{t+1}.– елемент "передбачення", що дозволяє будувати маршрут до цілі;

γ – коефіцієнт дисконтування, прийнятий рівним 0.9, що стимулює агента враховувати довгострокові перспективи (досягнення фінішу), а не лише миттєві бонуси.

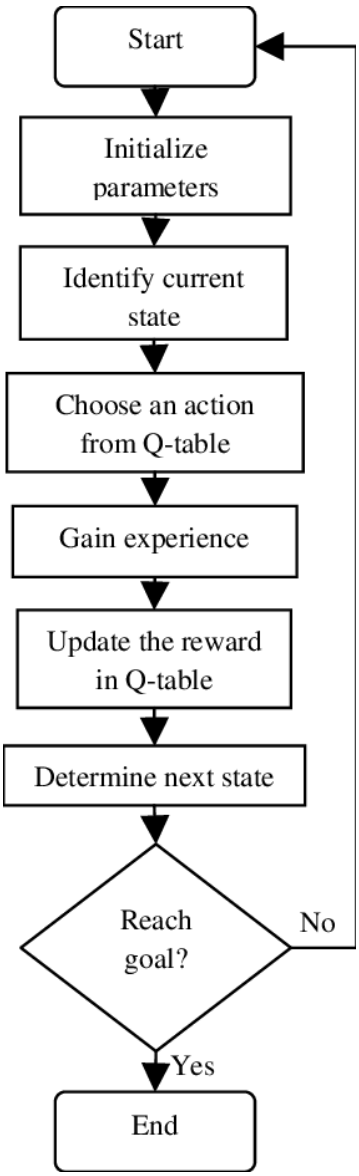


Рис. 2.3. Блок–схема ітераційного оновлення Q–значень

2.2.3. Система винагород

Час навчання критично залежить від коректного налаштування функції винагород. У даній роботі реалізовано наступну логіку формування винагороди r:

$$r = \begin{cases} +1000 \\ -100 \\ -1 \end{cases}$$

Штраф **–1** за кожен крок є важливим елементом оптимізації: він змушує агента шукати **найкоротший** шлях, оскільки кожен зайвий крок зменшує підсумкову сумарну винагороду. Великий штраф **–100** за зіткнення формує зону "небезпеки" навколо перешкод, яку агент вчиться уникати. **+1000,** якщо досягло цільової точки.

2.2.4. Стратегія вибору дій

Під час навчання агент стикається з дилемою "дослідження проти використання" (*Exploration vs. Exploitation*).

1. *Exploitation (Використання):* Агент обирає дію, яка має найбільше значення Q у поточній таблиці, тобто діє згідно з найкращим відомим планом;
2. *Exploration (Дослідження):* Агент обирає випадкову дію, щоб перевірити нові маршрути, які можуть виявитися кращими за відомі.

Для балансування цих підходів у роботі реалізовано є–жадібну стратегію Алгоритм вибору дії виглядає так:

1. Згенерувати випадкове число p є $[0,1]$;
2. Якщо $p <$ є, обрати випадкову дію a є A (дослідження) ;
3. Інакше, обрати дію з максимальним Q–значенням: $a = arg\ max_{a'} Q(s, a')$.

Параметр $epsilon$ динамічно змінюється протягом процесу навчання. На початку симуляції $\in = 1.0$ (повна випадковість дій для дослідження карти), і з кожним епізодом він лінійно зменшується до $\in_{min} = 0.1$. Це дозволяє агенту спочатку вивчити карту, а на пізніх етапах навчання –вдосконалювати знайдений оптимальний маршрут.

2.2.5. Алгоритмічна реалізація

Процес функціонування розробленого модуля можна описати наступним алгоритмом:

1. **Ініціалізація:**

– Завантаження карти середовища.
– Створення Q–таблиці розміром $H * W * 4$, заповненої нулями.
– Встановлення параметрів $a, \gamma,$ є.

2. **Цикл епізодів (навчання):**

Для кожного епізоду (від 1 до N):

1. Встановити агента у стартову точку s_{start}.
2. Поки не досягнуто фінішу або ліміту кроків:

– Обрати дію a згідно з ϵ –жадібною стратегією;
– Виконати дію, отримати новий стан s' та винагороду r;

– Оновити значення $Q(s, a)$ за рівнянням Беллмана;
– Перейти в новий стан: $s <- s'$.

3. Зменшити значення ϵ

Вилучення результату:

1. Після завершення навчання пройти маршрут від старту до фінішу, обираючи виключно дії з максимальним $Q(s, a)$ (жадібна стратегія) ;
2. Зберегти отриману послідовність дій як польотне завдання.

Математичний апарат гарантує збіжність алгоритму до оптимальної стратегії при достатній кількості епізодів навчання, [12] що робить його надійним інструментом для задач автономної навігації.

2.3. Формалізація задачі пошуку шляху в дискретному середовищі

Для реалізації алгоритмів управління автономним рухом БПЛА за допомогою обчислювальної техніки необхідно перейти від безперервного фізичного простору до його дискретної математичної моделі. Формалізація задачі пошуку шляху є критичним етапом проєктування, оскільки вона визначає, як саме програмний агент "бачить" навколишній світ, як він може з ним взаємодіяти та які критерії визначають успішність виконання польотного завдання. [14]

У рамках даної роботи використовується підхід, що базується на сіткових картах зайнятості (*Grid Occupancy Maps*). Цей метод дозволяє трансформувати складну топологію простору у матричну структуру, придатну для обробки алгоритмами навчання з підкріпленням.

2.3.1. Математична модель робочого простору

Нехай робочий простір польоту БПЛА представлено обмеженою двовимірною областю $W \cup R^2$. Для застосування чисельних методів здійснюється дискретизація області W шляхом накладання на неї регулярної сітки з кроком $\Delta\, d$, що відповідає фізичному розміру однієї комірки (наприклад, 1 метр).

Математичною моделлю середовища виступає матриця M розмірності $H \times W$ (де H – висота, W –ширина сітки в комірках). Кожен елемент матриці m_{ij} характеризує стан відповідної ділянки простору. Множина можливих значень комірки визначається як:

$$M = \{m_{ij} \mid i \in [0, H-1], j \in [0, W-1]\}$$

У розробленому програмному модулі, згідно з вхідними даними, прийнято наступну класифікацію типів комірок, що відображається множиною значень $V = \{0, 1, 2, 3\}$:

1. **Вільний простір (C_{free}):** $m_{ij} = 0$. Це ділянки, через які дозволено безперешкодний проліт БПЛА;
2. **Перешкоди (C_{obs}):** $m_{ij} = 1$. Це зони забороненого доступу. Фізично це означає, що якщо координати дрона потрапляють у цю комірку, фіксується аварійна ситуація (колізія);
3. **Точка старту (S_{start}):** $m_{ij} = 2$. Початкове положення агента;
4. **Цільова точка (S_{goal}):** $m_{ij} = 3$. Кінцева мета маршруту.

Таким чином, простір станів S для агента є дискретною множиною всіх пар координат (i, j), таких, що комірка не є перешкодою:

$$S = \{(i, j) \mid m_{ij} \neq 1\};$$

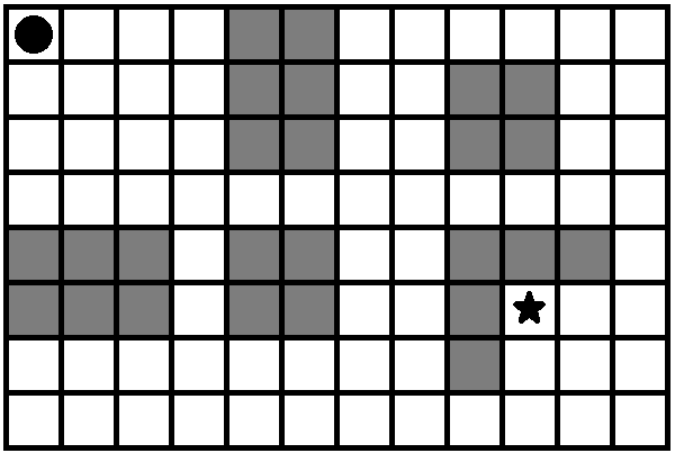


Рис. 2.4. Представлення робочого простору у вигляді сіткової карти зайнятості

2.3.2. Топологія зв’язності та кінематичні обмеження

При формалізації руху агента у дискретній сітці необхідно визначити правила переходу між станами. Це називається топологією зв'язності графа станів.

У даній роботі використовується **4–зв'язна топологія (окіл фон Неймана)**. Це означає, що з поточної комірки (i, j) агент може переміститися лише у сусідні комірки по вертикалі або горизонталі. Діагональні переміщення виключені для спрощення кінематичної моделі та підвищення безпеки маневрування (уникнення прольоту "крізь кут" перешкоди).

Множина допустимих дій A для агента у кожному такті часу t формалізується через вектори зміщення $(\Delta i, \Delta j)$:

$$\mathrm{A} = \begin{cases} \delta_0 : \Delta i = -1, \Delta j = 0 \\ \delta_1 : \Delta i = 0, \Delta j = +1 \\ \delta_2 : \Delta i = +1, \Delta j = 0 \\ \delta_3 : \Delta i = 0, \Delta j = -1 \end{cases}$$

Функція переходу $T(s, a)$, яка визначає новий стан s' після виконання дії a у стані $s = (i, j)$, описується системою рівнянь з урахуванням граничних умов:

$$s` = T(s,a) = \begin{cases} (i + \Delta i, j + \Delta j) \\ (i, j) \end{cases}$$

Друга умова системи відображає фізику зіткнення: якщо агент намагається переміститися у перешкоду або вилетіти за межі карти, його координати не змінюються, але в контексті алгоритму навчання (див. п. 2.2) він отримує негативну винагороду.

2.3.3. Визначення оптимального шляху

Задачу пошуку маршруту можна сформулювати як задачу пошуку оптимальної послідовності станів.

Нехай P –це впорядкована послідовність станів (траєкторія):

$$P = (s_0, s_1, s_2, \dots, s_k)$$

Маршрут P вважається **допустимим (*valid*)**, якщо виконуються наступні умови:

1. **Умова старту:** $s_0 = S_{start}$ (координати, де $m_{ij} = 2$).

2. **Умова фінішу:** $s_k = S_{goal}$ (координати, де $m_{ij} = 3$).
3. **Умова неперервності:** Для будь–якого $t[0, k-1]$, стан s_{t+1} є результатом допустимої дії із s_t.
4. **Умова безпеки:** Для будь–якого t, $s_t \epsilon\, C_{obs}$ (агент не проходить крізь перешкоди).

У рамках даної роботи використовується підхід, що базується на сіткових картах зайнятості (*Grid Occupancy Maps*). Цей метод, фундаментально обґрунтований у працях Ельфеса для задач навігації мобільних роботів[20], дозволяє трансформувати складну топологію простору у матричну структуру, придатну для обробки алгоритмами навчання з підкріпленням.

Оскільки простір дискретизовано рівномірною сіткою, а рух дозволено лише у 4 напрямках, відстань між точками у такому просторі визначається метрикою Манхеттена ($L_1 metric$):

$$D(A, B) = |x_A - x_B| + |y_A - y_B|$$

Метою роботи програмного модуля є знаходження такої траєкторії P^*, яка мінімізує кількість кроків k (довжину шляху). У термінах функції вартості $Cost(P)$, де вартість кожного переходу дорівнює 1 (що відповідає штрафу –1 у *Q–learning*):

$$P^* = arg\ min_P \sum_{t=0}^{k-1} Cost(s_t, s_{t+1})$$

Таким чином, задача зводиться до мінімізації кумулятивної вартості переходу від старту до цілі.

2.3.4. Переваги та обмеження дискретної моделі

Обраний підхід формалізації має ряд суттєвих переваг для реалізації на бортових обчислювачах:

1. **Передбачуваність пам'яті:** Розмір матриці M фіксований і відомий заздалегідь, що дозволяє статично виділяти пам'ять (як реалізовано у векторі *loadedMap*).

2. **Швидкість доступу:** Перевірка допустимості ходу ($O(1)$) зводиться до перевірки значення елемента масиву за індексом.
3. **Універсальність:** Будь–яка карта місцевості, отримана з супутника або лідара, може бути конвертована у бінарну сітку (перешкода/вільно).

Проте, модель накладає і певні обмеження, які необхідно враховувати при інтерпретації результатів:

1. **Апроксимація:** Реальна траєкторія польоту буде мати вигляд ламаної лінії з кутами 90 градусів. Для реального БПЛА це вимагатиме етапу згладжування траєкторії (*trajectory smoothing*) після отримання маршруту від *Q–learning* модуля;
2. **Проблема локальних мінімумів:** У складних лабіринтах прості жадібні алгоритми можуть застрягти, проте використання *Q–learning* з етапом розвідки ($\in$–*greedy*) дозволяє гарантовано знаходити вихід, якщо він існує.

Формалізація задачі у вигляді пошуку оптимального шляху на графі з 4–зв'язною топологією дозволяє безпосередньо застосувати табличний метод *Q–learning*. Матриця карти M стає основою для простору станів S, а вектори зміщення визначають простір дій A. Така математична модель є адекватною для поставленої мети дипломного проекту і забезпечує баланс між точністю навігації та обчислювальною складністю.

2.4. Постановка задачі розроблення програмного модуля

На основі проведеного аналізу методів машинного навчання (п. 2.1) та математичної формалізації середовища (п. 2.3), виникає необхідність у розробці спеціалізованого програмного забезпечення. Мета розробки створення програмного модуля управління польотом БПЛА, що реалізує алгоритм *Q–learning* для автономної навігації в умовах статичних перешкод.

Розробка програмного модулю вимагає чіткого визначення функціональних та нефункціональних вимог, структури вхідних і вихідних даних, а також обґрунтування вибору інструментальних засобів. [15]

2.4.1. Загальна характристика та призначення модуля

Проєктований програмний модуль призначений для інтеграції в систему управління безпілотним літальним апаратом. Його основним завданням є прийняття навігаційних рішень на основі попередньо завантаженої карти місцевості.

Модуль повинен працювати у двох основних режимах:

1. Режим навчання (*Training Mode*): Агент досліджує віртуальну копію середовища, заповнюючи таблицю цінності дій (*Q–table*) шляхом багаторазових симуляцій польоту;
2. Режим експлуатації (*Inference/Execution Mode*): Модуль використовує накопичені знання для побудови оптимального маршруту та генерації конкретних команд управління для контролера польоту.

2.4.2. Функціональні вимоги до програмного забезпечення

Виходячи з аналізу предметної області, до Модуля висуваються наступні функціональні вимоги:

1. Підсистема роботи з картографічними даними:

– Модуль повинен забезпечувати зчитування карти місцевості з текстових файлів;
– Необхідно реалізувати парсинг структури карти, де умовними позначеннями (числовими кодами) задані: вільний простір, перешкоди, точка зльоту та точка призначення;
– Система повинна проводити валідацію вхідних даних (перевірка на цілісність, наявність старту і фінішу, прямокутність матриці);
– Згідно з реалізацією: Пошук файлів карт повинен здійснюватися автоматично у робочій директорії та підпапках для зручності користувача.
– 2. Підсистема машинного навчання:
– Реалізація алгоритму Q – learning згідно з рівнянням Беллмана;
– Підтримка ϵ – жадібної стратегії вибору дій для балансу між дослідженням карти та використанням оптимального шляху;

– Можливість налаштування гіперпараметрів навчання: коефіцієнта навчання (α), коефіцієнта дисконтування (γ) та кількості епізодів.

3. Підсистема збереження та відновлення знань:

– Модуль повинен вміти серіалізувати (зберігати) навчену Q–таблицю у файл (Qtable.txt), щоб уникнути необхідності повторного навчання при кожному запуску;

– Реалізація механізму де серіалізації (завантаження) Q–таблиці для миттєвого початку роботи в режимі експлуатації.

4. Підсистема генерації управляючих команд:

– Критично важливою вимогою є перетворення абстрактного шляху (послідовності клітинок (x, y)) у набір команд, зрозумілих для польотного контролера;

– Агрегація команд: Модуль повинен об'єднувати послідовні кроки в одному напрямку. Наприклад, замість п'яти команд "Вперед на 1 метр", має генеруватися одна команда "Вперед на 5 метрів";

– Експорт команд у текстовий файл (*drone_commands.txt*) у форматі *COMMAND DISTANCE*.

2.4.3. Вимоги до вхідних та вихідних даних

Вхідні дані:

Основним вхідним потоком даних є файл топології карти (наприклад, map.txt). Формат файлу визначається як матриця цілих чисел, розділених пробілами, де:

1. 0 –дозволена зона польоту;
2. 1 –перешкода (стіна, дерево);
3. 2 –точка старту (*Start*);
4. 3 –цільова точка (*Goal*).

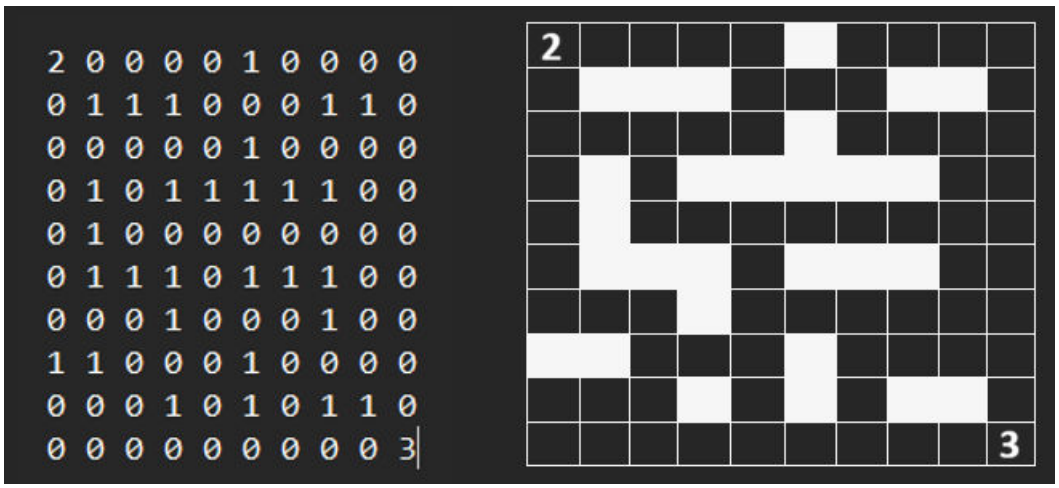


Рис. 2.5. Структура вхідних даних: текстовий формат та візуальне представлення

В результаті роботи Модуль повинен генерувати три типи артефактів:

1. Файл маршруту (*path.txt*): Містить "сирі" координати траєкторії руху агента. Використовується для візуалізації та налагодження;
2. Файл команд (*drone_commands.txt*): Містить інтерпретовані команди для автопілота.

Приклад:

1. *FORWARD 10.0, RIGHT 5.0.* ;
2. Саме цей файл є кінцевим продуктом роботи системи навігації;
3. Файл знань (*Qtable.txt*): Матриця ваг нейронних зв'язків (у даному випадку – значень *Q*–функції), що дозволяє переносити навчену модель на інші пристрої.

2.4.4. Нефункціональні вимоги та обмеження

Враховуючи специфіку застосування на борту БПЛА, до програмного забезпечення висуваються жорсткі вимоги щодо ефективності:

1. Швидкодія: Алгоритм прийняття рішення (вибір наступного кроку) у навченого агента повинен мати обчислювальну складність $O(1)$ (доступ до масиву), що забезпечує миттєву реакцію в реальному часі;

2. Автономність: Програма не повинна залежати від зовнішніх бібліотек машинного навчання (*TensorFlow* або *PyTorch*), які вимагають значних ресурсів пам'яті та складної інфраструктури. Реалізація має бути виконана на C++;
3. Кросплатформеність: Код повинен компілюватися як на настільних ОС (*Windows/Linux*) для навчання, так і на вбудованих системах (наприклад, *Raspberry Pi* або контролери на базі ARM), що використовуються в дронах;
4. Управління пам'яттю: Використання динамічних структур даних (*std::vector*) повинно бути оптимізоване для роботи з картами великої розмірності.

2.4.5. Обґрунтування вибору засобів розробки

Для реалізації поставленої задачі обрано мову програмування *C++* (стандарт *C++*17). Цей вибір зумовлений наступними факторами:

1. Висока продуктивність: *C++* забезпечує пряму компіляцію в машинний код без проміжних віртуальних машин, що є критичним для задач реального часу;
2. Керування ресурсами: Мова надає розробнику повний контроль над виділенням та звільненням пам'яті, що дозволяє уникнути непередбачуваних затримок (*lag spikes*), характерних для мов зі збирачами сміття (*Garbage Collection*), таких як *Java* або *Python*;
3. Бібліотека *STL*: Використання стандартної бібліотеки шаблонів (vector, fstream, filesystem) дозволяє реалізувати складні алгоритми обробки даних (сортування, пошук шляхів) без залучення сторонніх залежностей;
4. Сумісність з апаратним забезпеченням: Більшість сучасних *SDK* для дронів (*DJI SDK, PX4 Autopilot, ArduPilot*) написані на *C/C++,* що спрощує подальшу інтеграцію розробленого модуля.

Для написання коду та налагодження використовується середовище *Visual Studio* (або аналогічне), проте фінальний код повинен відповідати стандарту *ISO C++,* щоб забезпечити переносність на бортові комп'ютери під управлінням *Linux*.

Сформульована постановка задачі визначає розробку автономного програмного модуля на мові *C++,* який реалізує повний цикл навігації: від зчитування карти та

навчання методом *Q–learning* до генерації фізичних команд управління дроном. Відсутність залежностей від важких фреймворків та орієнтація на дискретну модель середовища дозволяють виконати вимоги щодо швидкодії та ефективності, необхідні для вбудованих систем БПЛА.

2.5. Висновки до розділу

У другому розділі дипломної роботи проведено комплексний теоретичний аналіз методів та підходів до створення систем автономного управління польотом безпілотних літальних апаратів. На основі дослідження предметної області обґрунтовано математичний апарат та сформульовано технічне завдання на розробку програмного модуля.

За результатами проведених досліджень можна зробити наступні висновки:

1. Аналіз методів машинного навчання показав, що для задач тактичної навігації в умовах невизначеності найбільш ефективним підходом є навчання з підкріпленням (*Reinforcement Learning*). На відміну від нейронних мереж (навчання з учителем), які вимагають великих розмічених наборів даних та значних обчислювальних потужностей, методи RL дозволяють агенту навчатися безпосередньо в процесі взаємодії із середовищем. Це робить їх ідеальними для імплементації на бортових системах БПЛА з обмеженим енергоресурсом;
2. Обґрунтовано вибір алгоритму *Q–learning* як базового методу для програмної реалізації. Його перевагами є безмодельність, здатність гарантовано знаходити оптимальний маршрут у дискретних середовищах та простота програмної реалізації. Розроблена математична модель, що включає рівняння Беллмана та $\in$–жадібну стратегію вибору дій, забезпечує баланс між дослідженням карти та використанням знайдених оптимальних шляхів. Спроектована система винагород (штраф за крок, значний штраф за зіткнення, винагорода за ціль)

математично гарантує пошук найкоротшого та найбезпечнішого маршруту;

3. Виконано формалізацію задачі пошуку шляху. Реальний фізичний простір польоту успішно трансформовано у математичну модель у вигляді сіткової карти зайнятості (*Grid Map*). Визначено множину станів (координати дрона) та множину допустимих дій (рух у чотирьох напрямках). Такий підхід дозволив звести складну навігаційну задачу до задачі пошуку оптимальної послідовності станів на графі, що суттєво спрощує алгоритмічну реалізацію;
4. Сформульовано постановку задачі розроблення програмного модуля. Визначено чіткі вимоги до вхідних даних (текстові карти топології) та вихідних артефактів (файли маршрутів та команд управління). Встановлено, що розробка вестиметься мовою *C++* з використанням стандартної бібліотеки (*STL*), що забезпечить високу швидкодію, кросплатформеність та незалежність від сторонніх фреймворків. Ключовою вимогою визначено здатність модуля генерувати агреговані команди (наприклад, *FORWARD* 10.0) для прямої передачі на польотний контролер.

Таким чином, у даному розділі створено необхідний теоретичний та методологічний фундамент для практичної частини роботи. Обраний математичний апарат (*Q–learning*) та інструментальні засоби (*C++*) повністю відповідають меті роботи і дозволяють перейти до безпосереднього проектування та програмної реалізації модуля управління польотом, що буде детально розглянуто у наступному розділі.

РОЗДІЛ 3
ПРОВЕКТУВАННЯ ПРОГРАММНОГО МОДУЛЯ

3.1. Розроблення сценаріїв управління польотом БПЛА з машинним навчанням

Проєктування програмного модуля починається з визначення сценаріїв його використання (*Use Cases*). Це дозволяє формалізувати взаємодію між користувачем (оператором БПЛА) та системою (програмним модулем *ML*), а також чітко окреслити межі функціональності.

Враховуючи специфіку розроблюваної системи, яка базується на методі навчання з підкріпленням (*Q–learning*), виділяються два основні етапи роботи: етап навчання (*Training*) та етап експлуатації (*Inference*).

3.1.1. Ідентифікація акторів та претедентів

Головним актором системи є Оператор. Це особа, яка відповідає за завантаження польотного завдання та ініціалізацію процесу розрахунку маршруту.

Система розглядається як "чорна скринька", що надає певний набір функцій. На основі аналізу функціональних вимог (розділ 2.4) та реалізованого інтерфейсу (меню програми), виділено наступні прецеденти (*Use Cases*):

1. Завантаження карти середовища: Оператор обирає файл топології місцевості;
2. Навчання моделі (*ML Training*): Запуск алгоритму *Q–learning* для генерації оптимальної політики поведінки;
3. Симуляція польоту: Візуалізація руху дрона на основі отриманих знань;
4. Експорт польотних команд: Генерація файлу з фізичними командами управління.

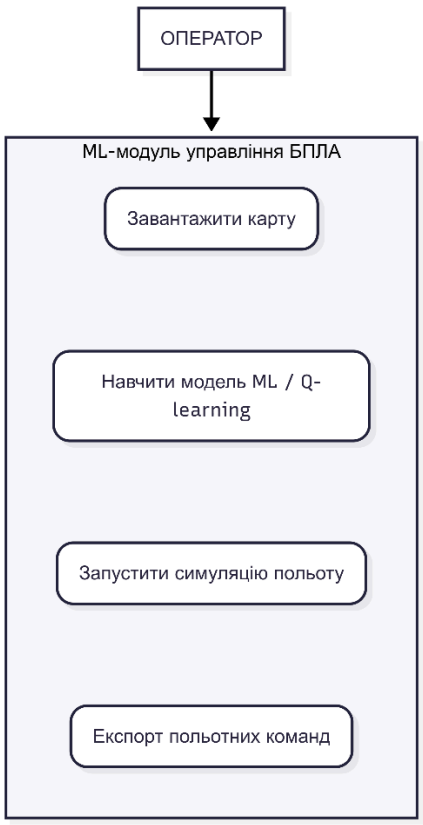


Рис. 3.1. Діаграма прецедентів програмного модуля

3.1.2. Опис основних сценаріїв взаємодії

Подивимось на детальний потік подій для ключового сценарію –**"Автономне планування маршруту"**, він об'єднує навчання та генерацію шляху.

Сценарій: Отримання маршруту в незнайомому середовищі

Передумова: Система запущена, файли карт знаходяться у робочій директорії.

Основний потік подій:

1. Оператор обирає пункт меню "Завантажити карту".
2. Система сканує директорію, виводить список доступних файлів (*map*.txt*) і очікує вибору.
3. Оператор вводить номер файлу.
4. Система зчитує матрицю, валідує наявність точок старту та фінішу.
5. Оператор обирає пункт "Обучити модель".
6. Система ініціалізує *Q*–таблицю і запускає цикл епізодів. В консоль виводиться прогрес (номер епізоду та поточний $\in$).

7. Після завершення навчання система зберігає отримані знання (Qtable.txt).
8. Оператор обирає "Запустити симуляцію" або "Експорт команд".
9. Система проходить маршрут, використовуючи "жадібну" стратегію, і записує файл drone_commands.txt.

3.1.3 Моделювання потоку діяльності

Для проектування логіки переходів між станами програми використовується діаграма діяльності (*Activity*). Оскільки програмний модуль реалізовано у вигляді консольного додатка з циклічним меню (функція *main* та *printMenu*), потік управління є лінійним з розгалуженням на вибір користувача.

Алгоритм роботи програми на рівні користувацького сценарію виглядає наступним чином:

1. Ініціалізація: Запуск програми, встановлення генератора випадкових чисел.
2. Цикл обробки подій: Програма очікує вводу команди;
3. Розгалуження (*Switch*): Залежно від вибору (1–6) викликається відповідна підпрограма;
4. Виконання: Наприклад, при виборі навчання виконується функція *trainQLearning*();
5. Повернення: Після виконання функції управління повертається в головне меню.

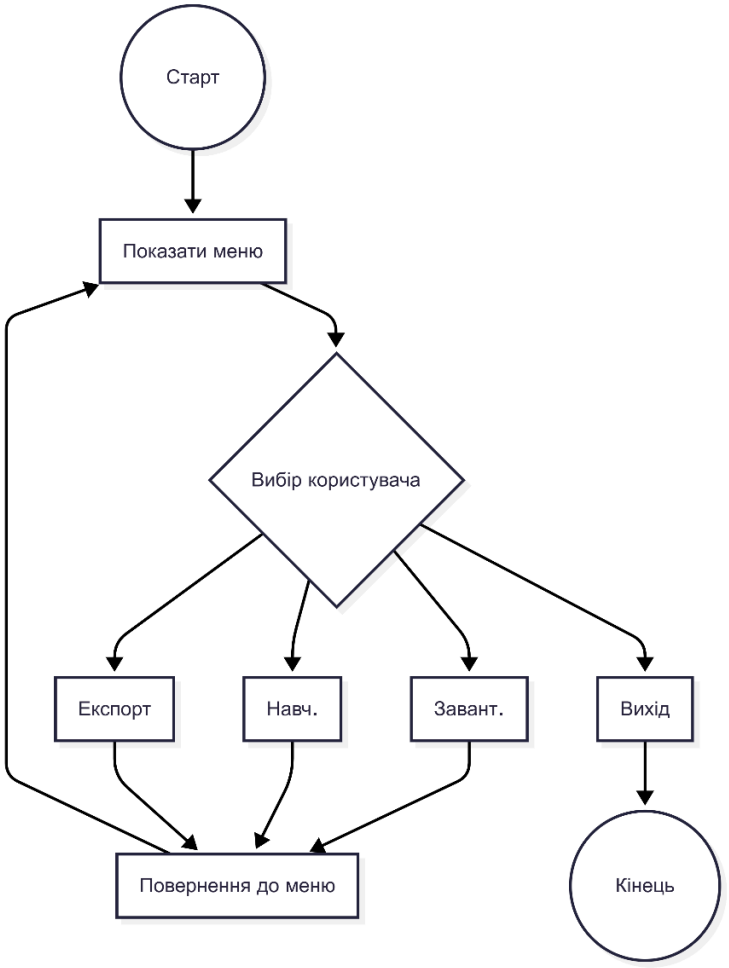


Рис. 3.2. Діаграма діяльності загального алгоритму роботи

3.1.4. Схема інформації потоків

Важливою частиною сценарію є рух даних у системі. У розробленому модулі реалізовано конвеєрний принцип обробки даних (*Data Pipeline*).

1. *Input* (Вхід): "Сира" карта (матриця чисел).
2. *Processing* (Обробка ML): Трансформація карти у Q–таблицю (матрицю ймовірностей/ваг).
3. *Post–processing* (Пост–обробка): Трансформація Q–таблиці у список координат (шлях).
4. *Output* (Вихід): Трансформація шляху у векторні команди (дистанція/напрямок).

Графічно цей процес представлено на структурній схемі сценарію обробки даних:

Рис. 3.3. Схема потоку даних у сценарії управління

Цей сценарій дозволяє забезпечити повну простежуваність дій: від вхідного файлу до реального руху дрона, що є критичним для систем безпеки польотів.

3.2. Статична структура програмного модуля

Статична структура програмного забезпечення визначає організацію програмного коду, розподіл функціональності між окремими компонентами та ієрархію даних. Для розроблюваного модуля управління БПЛА обрано модульну архітектуру на базі процедурного підходу мови *C++*. Це дозволяє мінімізувати накладні витрати пам'яті (*overhead*), характерні для об'єктно–орієнтованих структур, що є критичним для вбудованих систем.

Програмний засіб складається з трьох логічних рівнів:

1. Рівень даних (*Data Layer*): Глобальні структури для зберігання топології карти та знань агента;
2. Рівень бізнес–логіки (*Core Logic*): Функції реалізації алгоритму *Q–learning* та навігації;
3. Рівень інтерфейсу та вводу–виводу (*IO/UI Layer*): Функції взаємодії з користувачем та файловою системою.

3.2.1. Структурна схема компонентів

Логічна архітектура програми може бути представлена у вигляді взаємопов'язаних функціональних блоків. Центральним елементом є керуючий блок (функція *main*), який диспетчеризує виклики до спеціалізованих підпрограм.

Відокремлення логіки завантаження карт (*File Loader*) від логіки навчання (*ML Engine*) дозволяє легко змінювати формат вхідних даних без переписування алгоритму навчання.

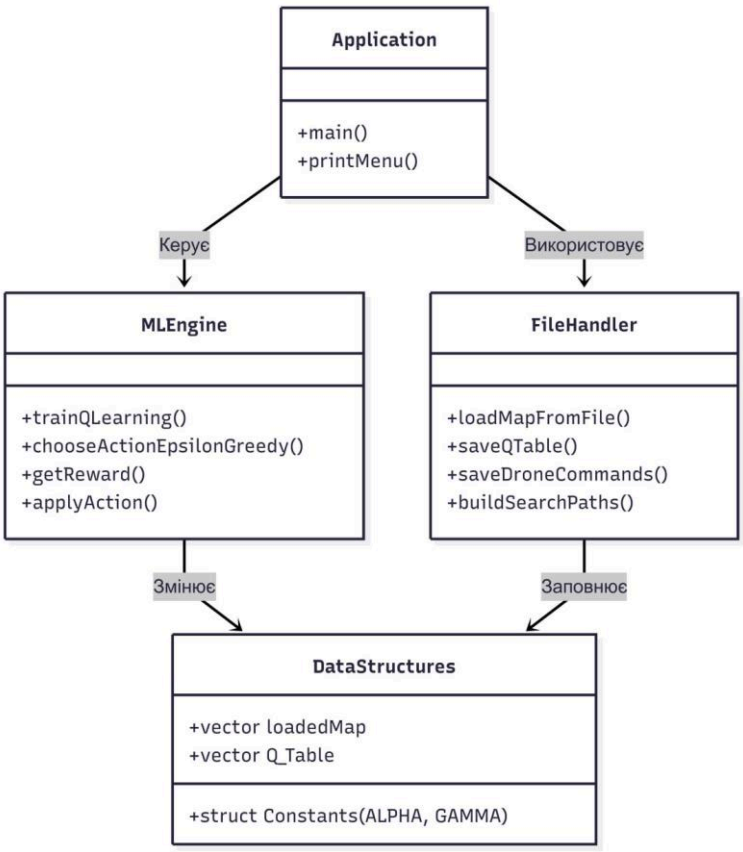


Рис. 3.4. Компонентна діаграма статичної структури модуля

3.2.2. Організація структур даних

Ефективність роботи алгоритму *Q–learning* напряму залежить від способу організації пам'яті. У розробленому модулі використовуються дві ключові динамічні структури даних, реалізовані через стандартний контейнер *std::vector* бібліотеки *STL*.

Матриця середовища (*loadedMap*).

Двовимірний масив цілих чисел *vector*<*vector*<*int*>> де:

1. Індексація: *loadedMap*[*row*][col].

2. Семантика значень: 0 (пусто), 1 (стіна), 2 (старт), 3 (фініш).

3. Ця структура є статичною під час виконання епізоду (карта не змінюється).

Таблиця знань (*Q table*).

Двохвимірна структура, що зберігає ваги ("цінність") для кожної можливої дії у кожній точці простору. У коді це реалізовано як: *vector*<*array*<*double*, 4>>> де:

1. Виміри: Висота карт × Ширина карти × 4 дії.

2. Тип даних *double* забезпечує необхідну точність для градієнтних змін винагороди.

Графічно зв'язок між картою та *Q*–таблицею можна представити як нашарування вектора дій на кожну клітинку простору.

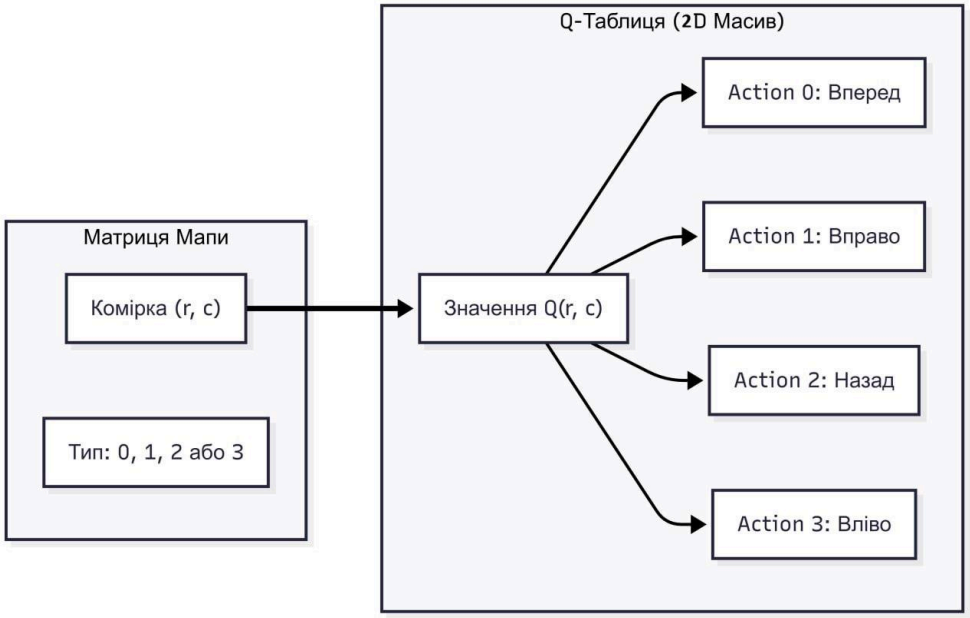


Рис. 3.5. Схема організації даних: відповідність карти та *Q*–таблиці

3.2.3. Функціональна декомпозиція

Програмний код розбито на окремі функції, кожна з яких виконує атомарну задачу. Такий підхід спрощує налагодження та тестування. Нижче наведено опис основних функціональних блоків:

Блок А. Робота з файловою системою:

1. *loadMapFromFile*(*string filename*): Зчитує текстовий файл, парсить рядки та формує матрицю *loadedMap*. Перевіряє валідність (прямокутність) карти;
2. *saveQTable*(...) / *loadQTable*(...): Забезпечують персистентність (збереження) навченої моделі;
3. *saveDroneCommands*(...): Ключова функція для перетворення шляху в інструкції. Реалізує алгоритм агрегації (*RLE*–подібне стиснення), об'єднуючи послідовні кроки в одну команду.

Блок Б. Логіка машинного навчання (*ML Core*):

1. *trainQLearning*(): Основний цикл навчання. Керує епохами та зменшенням параметру ∈;
2. *chooseActionEpsilonGreedy*(...): Реалізує стохастичну політику вибору дій;
3. *getReward*(...): Функція зворотного зв'язку, що повертає числові значення (+1000, –100, –1) залежно від наслідків дії;
4. *applyAction*(...): Детермінована модель кінематики дрона (обчислення нових координат).

Блок В. Допоміжні утиліти:

1. *findStartGoal*(...): Сканує карту для пошуку точок входу та виходу;
2. *printMapWithPath*(...): Візуалізатор, що накладає знайдений шлях на карту символами *ASCII*.

3.2.4. Взаємозв’язки модулів

Аналіз розробленої програмної системи дозволяє виділити чітку ієрархічну взаємодію між компонентами. Статична структура визначає суворі односпрямовані залежності між модулями, що забезпечує прогнозовану поведінку програми та спрощує її супровід. У даній реалізації принципово відсутня циклічна залежність (*circular dependency*), яка часто є джерелом помилок при компіляції та виконанні складних проєктів мовою *C++*.

Головний модуль *Main* виступає у ролі оркестратора системи. Він ініціює роботу інших компонентів, керуючи їх життєвим циклом, але не втручається в їхню внутрішню логіку. При цьому модуль машинного навчання (*ML Engine*) виступає виключно споживачем даних: він використовує інформацію з модуля Карти, але не має зворотного впливу на нього. Це гарантує цілісність вихідних даних про середовище протягом усього процесу навчання.

Така архітектура реалізує принцип слабкої зв’язності (*Low Coupling*). Модуль *Q–learning* оперує лише математичною абстракцією матриці перешкод і не містить коду для роботи з файловою системою. Це має важливе практичне значення для масштабування та модернізації проєкту.

3.3. Динаміка функціонування програмного модуля

Динаміка функціонування програмного модуля визначає послідовність зміни станів системи під час виконання алгоритмів навчання та навігації. Оскільки модуль реалізує ітеративний алгоритм *Q–learning*, його робота базується на циклічній взаємодії агента з віртуальним середовищем. Розглянемо детально динамічні процеси, що протікають у системі.

3.3.1. Алгоритмічний цикл навчання

Основним динамічним процесом є цикл навчання, реалізований у функції *trainQLearning*(). Цей процес є вкладеним і складається з двох рівнів ітерацій: зовнішнього (епізоди) та внутрішнього (кроки або такти симуляції).

Динаміку одного епізоду навчання можна описати наступним алгоритмом:

1. Ініціалізація стану: Агент розміщується у точці старту;
2. Вибір дії: Система приймає рішення про наступний крок на основі поточної політики (функція *chooseActionEpsilonGreedy*);
3. Взаємодія із середовищем: Розраховуються нові координати. Перевіряється валідність клітинки (чи не є вона першкодою);
4. Оцінка наслідків: Обчислюється миттєва винагорода;
5. Оновлення знань: Відбувається модифікація комірки Q–таблиці згідно з отриманим досвідом;
6. Перехід: Агент переміщується у новий стан, якщо не було колізії.

Цей процес повторюється до тих пір, поки агент не досягне цілі або не вичерпає ліміт кроків (*MAX_STEPS_PER_EPISODE*).

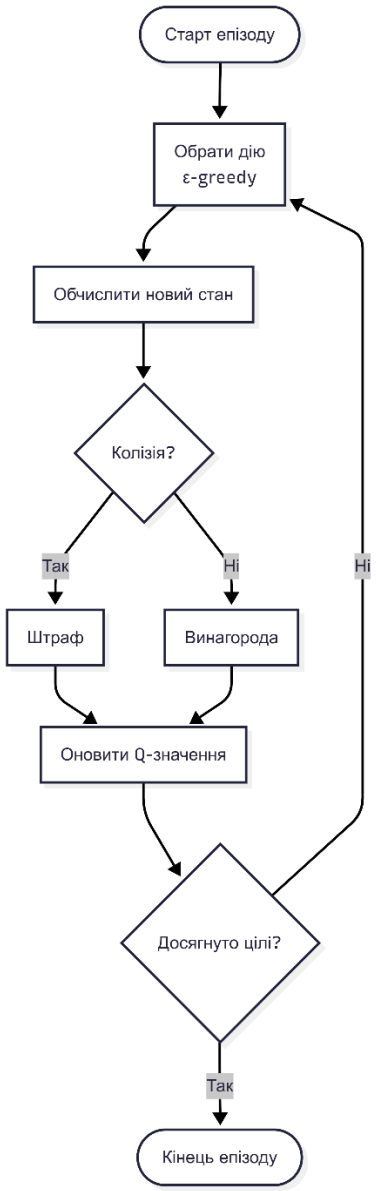


Рис. 3.6. Блок–схема динамічного циклу навчання

3.3.2. Динаміка станів агента

З точки зору теорії автоматів, модуль функціонує як скінченний автомат (*Finite State Machine*), де станами є координати (r, c) у матриці карти.

Переходи між станами ініціюються вхідними сигналами (діями $a \in \{0, 1, 2, 3\}$). Важливою особливістю реалізованої динаміки є обробка колізій:

1. Якщо дія веде у вільну клітинку ($m_{ij} = 0$), стан змінюється: $S_t \to S_{t+1}$.

2. Якщо дія веде у перешкоду ($m_{ij} = 1$), відбувається "відскок": стан залишається незмінним $S_t \to S_t$, але агент отримує негативну винагороду (штраф).

Така динаміка емулює фізичне зіткнення дрона зі стіною, змушуючи алгоритм уникати таких дій у майбутньому.

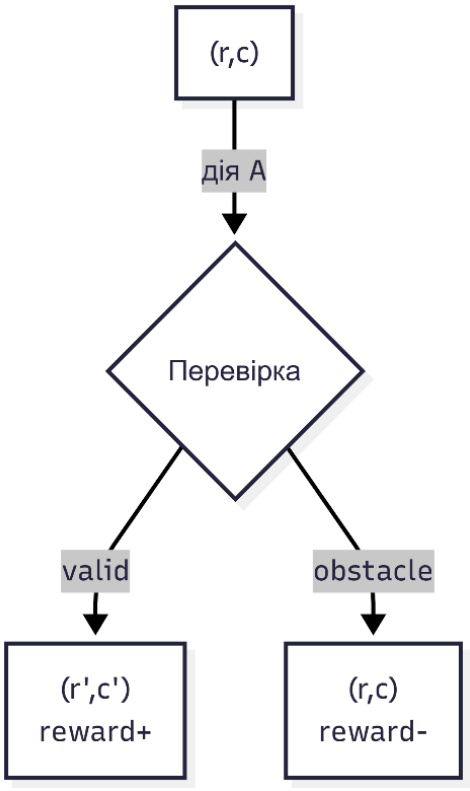


Рис. 3.7. Діаграма переходів станів при взаємодії з перешкодами

3.3.3. Часова еволюція параметрів навчання

Динаміка системи також характеризується зміною внутрішніх параметрів у часі, зокрема коефіцієнта розвідки ∈ (*epsilon*). На початку роботи (*Episode* = 1) система перебуває у стані **"Дослідження"** (*Exploration*), де ∈ ≈ 1.0. Агент рухається хаотично, скануючи топологію карти. З плином часу (лінійне зменшення у кожному епізоді) система плавно переходить у стан **"Використання"** (*Exploitation*).

Математично динаміка зміни описується рівнянням, реалізованим у коді:

$$\epsilon_{t+1} = max(\epsilon_{end}, \epsilon_t - \Delta\epsilon)$$

Де $\Delta_\epsilon = \frac{\epsilon_{start} - \epsilon_{end}}{N_{episode}}$. Це забезпечує збіжність алгоритму до стабільної траєкторії на фінальних етапах роботи.

3.3.4. Послідовність операцій при формуванні польотного завдання

Після завершення навчання активується динамічний процес вилучення результатів (функція *extractGreedyPathFromQ* та *saveDroneCommands*). Цей процес є лінійним і не містить стохастичних (випадкових) елементів. Його можна представити діаграмою послідовності, яка демонструє взаємодію основних блоків програми.

1. Головний модуль запитує оптимальний шлях;
2. Модуль ML проходить по карті, обираючи дії з max *Q*;
3. Отриманий список координат передається у конвертер команд;
4. Конвертер агрегує кроки (наприклад, 5 кроків "Вперед" "*FORWARD* 5") і записує файл.

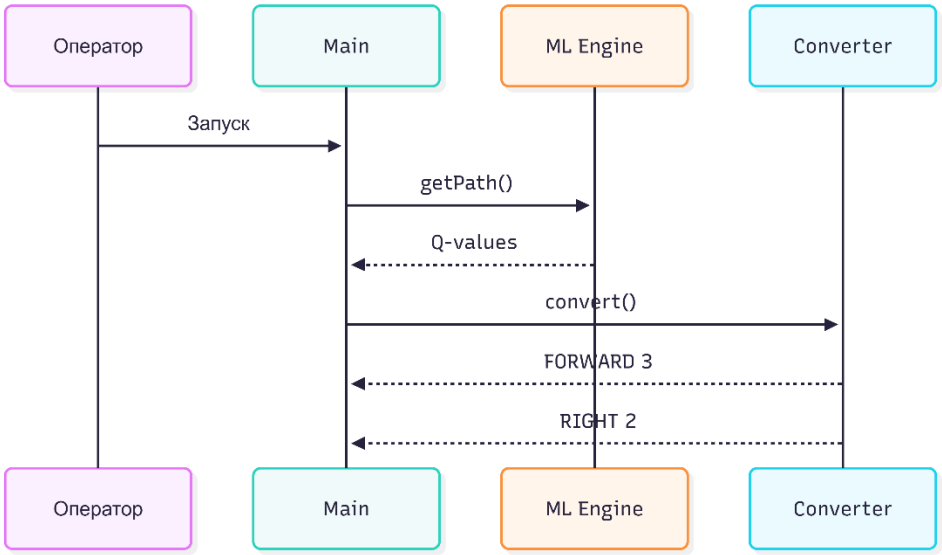


Рис. 3.8. Діаграма послідовності формування команд управління

3.4. Алгоритм управління з використанням машинного навчання

Розроблений алгоритм управління базується на методі *Q–learning*, який дозволяє автономному агенту (БПЛА) формувати оптимальну стратегію поведінки

шляхом багаторазової взаємодії з симульованим середовищем. Процес управління розділено на два етапи: навчання (*Training*), де формується база знань, та виконання (*Execution*), де знання трансформуються у керуючі команди.

3.4.1. Формальний опис алгоритму навчання

Алгоритм навчання реалізовано як ітераційну процедуру. На вхід подається карта місцевості M та параметри навчання (a,γ,∈). Виходом алгоритму є заповнена матриця цінності дій Q .

Нижче наведено псевдокод алгоритму, що відображає логіку функції *trainQLearning*:

```
initialize Q_Table[s][a] = 0 for all states s and actions a;

for (int e = 1; e <= Episodes; e++) {
    State S = Start_Point;
    int Steps = 0;

    while (S != Goal_Point && Steps < Max_Steps) {

        double p = random(0.0, 1.0);
        Action A;

        if (p < Epsilon) {
            A = randomAction();
        }
        else {
            A = argmax(Q[S][a]);
        }

        State S_next = applyAction(S, A);
        int R;

        if (isObstacle(S_next) || outOfBounds(S_next)) {
            R = -100;
            S_next = S;
        }
        else if (S_next == Goal_Point) {
            R = +1000;
        }
        else {
            R = -1;
        }

        Q[S][A] = Q[S][A] +
            Alpha * (R + Gamma * max(Q[S_next][a]) - Q[S][A]);

        S = S_next;
        Steps++;
    }

    Epsilon = decay(Epsilon);
}
```

Рис. 3.9. Навчання агента *Q–Learning*

3.4.2. Блок –схема ядра алгоритму

Ключовим моментом алгоритму є блок прийняття рішення та оновлення ваг. На відміну від класичного програмування, де поведінка жорстко задана правилами if–then, тут поведінка еволюціонує.

Графічно логіку одного кроку навчання представлено на блок–схемі алгоритму. Схема демонструє розгалуження при виборі дії та логіку обробки зіткнень.

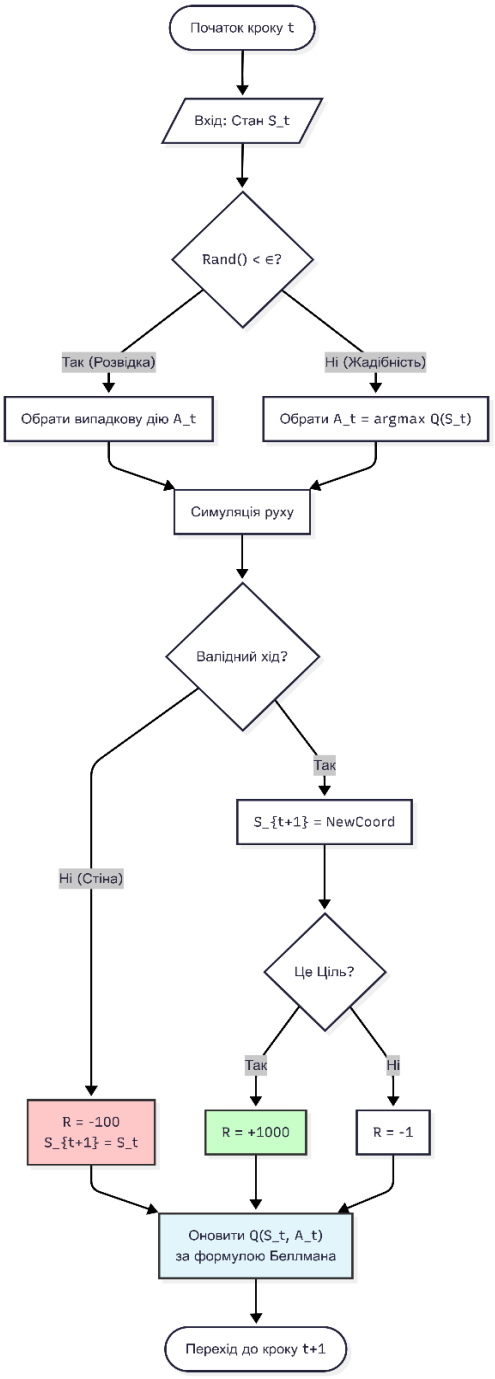


Рис 3.10. Блок–схема оновлення Q–значень

3.4.3. Алгоритм генерації керуючих команд

Після завершення навчання виконується алгоритм екстракції шляху. Особливістю даної реалізації є агрегація атомарних переміщень. Так, як *Q–learning* працює з дискретними кроками (1 клітинка = 1 метр), "сирий" вихід алгоритму виглядає як набір однотипних дій. Для оптимізації роботи контролера польоту застосовується алгоритм згортання.

Логіка агрегації команд:

1. Ініціалізувати порожній список команд;
2. Пройти по знайденому шляху $P = \{p_0, p_1, \dots, p_k\}$;
3. Визначити вектор руху $v = p_i - p_{i-1}$;
4. Якщо поточний вектор v співпадає з попереднім вектором руху:

– Збільшити лічильник дистанції поточниої команд($dist \leftarrow dist + 1.0$).

5. Інакше:

– Записати попередню команду у файл;

– Створити нову команду на основі вектора v.

6. Записати фінальну команду.

– Цей підхід дозволяє зменшити кількість команд, що передаються на дрон, у 5–10 разів, забезпечуючи більш плавний політ;

– Приклад роботи алгоритму агрегації:

7. **Вхід (з Q–таблиці):** Вперед, Вперед, Вперед, Вправо, Вправо.
8. **Вихід (у файл):**

– FORWARD 3.0;

– RIGHT 2.0.

3.5. Приклад програмної реалізації управління польотом БПЛА

Програмна реалізація розробленого модуля виконана мовою програмування *C++* (*ISO C++17*). Вибір даного стандарту обумовлений необхідністю використання

сучасних засобів роботи з файловою системою (<*filesystem*>) для автоматичного пошуку карт, а також контейнерів *STL* для ефективного управління пам'яттю.

3.5.1. Структура проекту та середовище розробки

Розробка велася у середовищі *Microsoft Visual Studio*. Проект реалізовано у вигляді консольного застосунку, що забезпечує максимальну сумісність та простоту портування на вбудовані системи (наприклад, *Raspberry Pi* з *ОС Linux*) в діючі БПЛА та схожі системи.

Файлова структура проекту включає:

1. *main.cpp* — основний файл з точкою входу та реалізацією логіки *Q–learning*;
2. *maps*/ — директорія, що містить топологічні карти у форматі *.txt*;
3. *drone_commands.txt* — вихідний файл з генерованими інструкціями.

3.5.2. Реалізація інтерфейсу користувача

Для взаємодії з оператором розроблено текстовий інтерфейс (CLI), який працює у циклічному режимі. Головне меню надає доступ до всіх функцій системи.

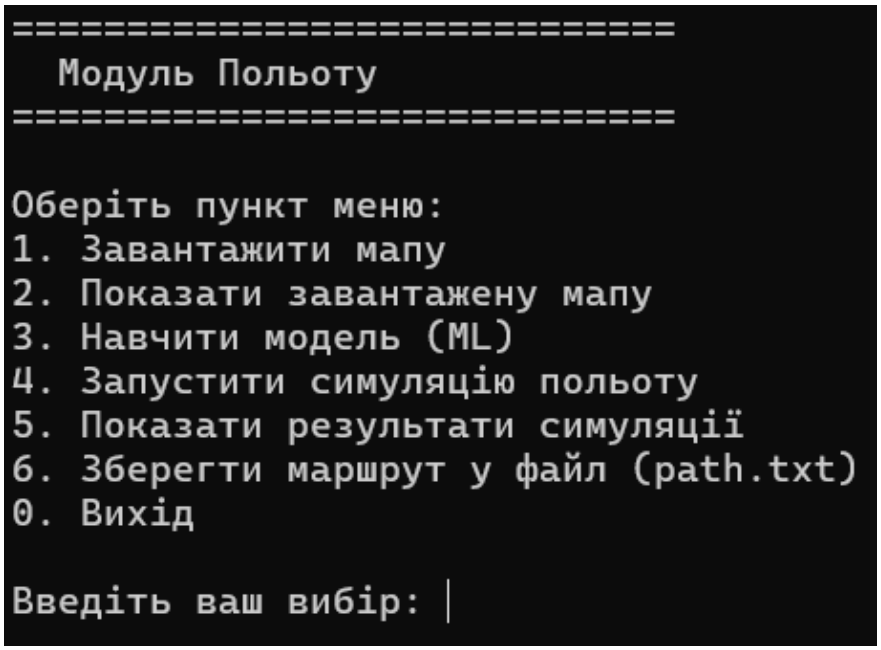


Рис. 3.11. Головне меню програми

3.5.3. Програмна реалізація ядра навчання

Ключовим компонентом системи є функція оновлення Q–таблиці. Нижче наведено лістинг коду, що реалізує рівняння Беллмана. У реалізації використано тривимірний вектор *vector*<*array*<*double*, 4>>> *Q* для зберігання ваг.

```
// Отримання винагороди за дію
double reward = getReward(next.first, next.second, goal, bumped);

// Отримання поточного значення Q
double oldQ = Q[r][c][action];

// Пошук максимального Q для наступного стану (Max Q')
double maxNextQ = -numeric_limits<double>::infinity();
for (int a = 0; a < ACTIONS; ++a) {
    maxNextQ = max(maxNextQ, Q[next.first][next.second][a]);
}

// Формула Беллмана: Q_new = Q_old + alpha * (Reward + gamma * MaxQ' - Q_old)
Q[r][c][action] = oldQ + ALPHA * (reward + GAMMA * maxNextQ - oldQ);
```

Рис. 3.12. Реалізація кроку навчання

Даний фрагмент демонструє безпосереднє перенесення математичної моделі у програмний код. Константі параметри *ALPHA* (0.1) та *GAMMA* (0.9) задані як глобальні константи, що дозволяє легко налаштовувати динаміку навчання.

3.5.4. Демонстрація роботи симулятора

Для перевірки працездатності було проведено серію експериментів.

Крок 1. Завантаження карти. Система зчитує файл *map1.txt*, що містить складний лабіринт з перешкодами (позначені як 1).

Крок 2. Процес навчання. Запущено навчання на 5000 епізодів. У консоль виводиться лог процесу, де видно поступове зменшення параметру epsilon (від 1.0 до 0.1), що свідчить про перехід від стратегії розвідки до стратегії використання.

Крок 3. Результат планування. Після завершення навчання система генерує маршрут. Нижче наведено візуалізацію прокладеного шляху на карті, де символом * позначено траєкторію руху дрона від старту (*S*) до цілі (*G*).

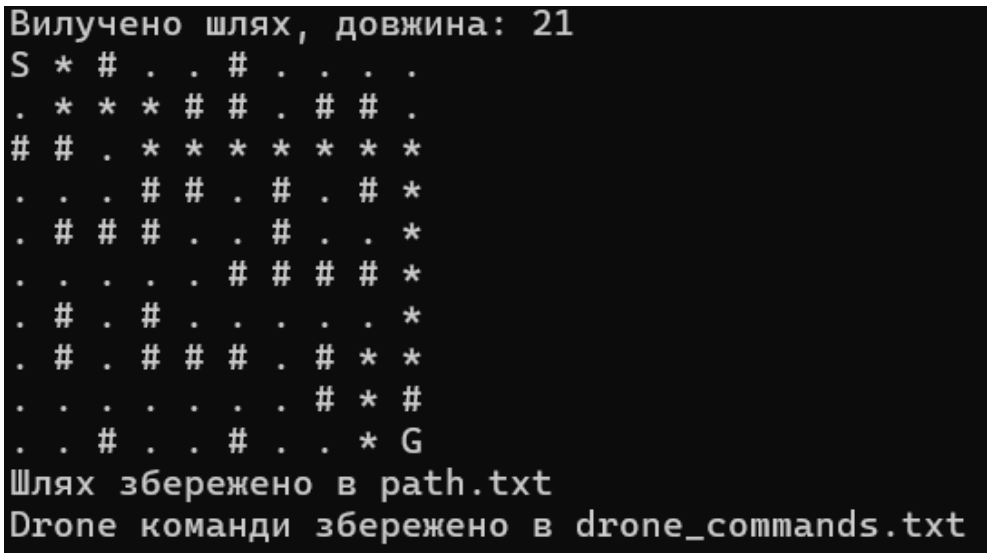


Рис. 3.13. Візуалізація знайденого оптимального маршруту

3.5.5. Генерація вихідних даних для контролера

Фінальним етапом роботи модуля є генерація файлу команд. Функція *saveDroneCommands* виконує агрегацію послідовних кроків. Наприклад, для маршруту, зображеного на (Рис. 3.13), програмний модуль згенерував наступний файл управління *drone_commands.txt*:

```
# Drone commands (2D). Units: meters
# Format: COMMAND DISTANCE


RIGHT 1
BACK 1
RIGHT 2
BACK 1
RIGHT 6
BACK 5
LEFT 1
BACK 2
RIGHT 1
```

Рис. 3.14. Вміст згенерованого файлу команд

Цей файл є готовим до завантаження у польотний контролер дрона (наприклад, на базі ArduPilot або ROS–ноди). Агрегація команд дозволила зменшити кількість

інструкцій з 20 (кількість клітинок шляху) до 9 (кількість лінійних сегментів), що оптимізує політ.

3.6. Висновки до розділу

У третьому розділі роботи виконано комплексне проектування та програмну реалізацію модуля управління польотом БПЛА з використанням методів машинного навчання. На основі теоретичних засад, розглянутих у попередніх розділах, було розроблено повнофункціональне програмне забезпечення, що вирішує задачі автономної навігації в дискретному середовищі.

За результатами проектування можна зробити такі висновки:

1. Розроблено архітектуру програмного забезпечення. Визначено та обґрунтовано статичну структуру модуля, яка базується на модульному принципі з чітким розмежуванням рівнів даних, бізнес–логіки та інтерфейсу користувача. Вибір мови програмування C++ та відмова від використання громіздких зовнішніх бібліотек *ML* на користь нативної реалізації (з використанням *STL*) дозволили забезпечити високу продуктивність системи та її портативність для потенційного використання на вбудованих обчислювачах (бортових комп'ютерах дронів).
2. Формалізовано сценарії та динаміку роботи. За допомогою мови моделювання *UML* (діаграми прецедентів, діяльності та послідовності) детально описано логіку функціонування системи. Виділено два ключові режими роботи: режим навчання, де відбувається наповнення бази знань (Q–таблиці), та режим експлуатації, де система генерує оптимальні маршрути. Показано, що динаміка навчання з використанням –жадібної стратегії забезпечує поступовий перехід від дослідження середовища до використання оптимальних стратегій.
3. Реалізовано алгоритмічне ядро системи. Програмно втілено математичну модель *Q–learning*. Алгоритм коректно обробляє взаємодію агента з

перешкодами, формує систему винагород та штрафів, що гарантує збіжність до оптимального маршруту. Реалізована система збереження знань (серіалізація Q–таблиці) дозволяє використовувати навчену модель багаторазово без необхідності повторного навчання, що є критичним для оперативного застосування БПЛА.

4. Створено механізм інтеграції з апаратною частиною. Важливим практичним результатом розділу є реалізація алгоритму конвертації та агрегації маршруту. Розроблена функція перетворює дискретну послідовність координат у набір векторних команд (*FORWARD*, *RIGHT* тощо) із зазначенням точної дистанції. Це вирішує проблему надлишковості команд і дозволяє передавати сформований файл drone_commands.txt безпосередньо на вхід польотного контролера.
5. Підтверджено працездатність розробки. Наведеним прикладом програмної реалізації продемонстровано, що створений модуль успішно зчитує топологію карти, навчається уникати перешкод і прокладає найкоротший шлях до цілі. Інтерфейс користувача забезпечує зручне керування процесом, а система логування дозволяє оператору контролювати хід навчання.

Таким чином, у даному розділі повністю вирішено задачу створення програмного модуля управління. Розроблена система є готовим інженерним рішенням, яке поєднує алгоритмів навчання з підкріпленням та надійність архітектури *C++*, і може бути використана як основа для створення автономних навігаційних комплексів БПЛА.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМНОГО МОДУЛЯ

4.1. Стратегія тестування та підготовка вихідних даних

Етап тестування є невід'ємною складовою життєвого циклу розробки програмного забезпечення. Його основна мета –гарантувати, що розроблений модуль управління польотом БПЛА функціонує згідно з вимогами, визначеними у технічному завданні, а реалізовані алгоритми машинного навчання забезпечують знаходження оптимального та безпечного маршруту.

Враховуючи специфіку роботи систем штучного інтелекту, процес тестування розділено на дві складові: перевірка програмної логіки (чи працює код без помилок) та валідація моделі машинного навчання (чи дійсно агент навчається).

4.1.1. Вибір стратегії тестування

Для всебічної оцінки якості програмного продукту обрано комбіновану стратегію, що поєднує методи «чорної скриньки» та «білої скриньки».

1. **Тестування «Чорної скриньки» (*Black–Box Testing*):** Зосереджується на перевірці вхідних та вихідних даних без аналізу внутрішньої структури коду.
   1) *Об'єкт перевірки:* Інтерфейс користувача (*CLI*), механізми завантаження файлів карт, коректність формату вихідного файлу drone_commands.txt;
   2) *Критерій успіху:* Програма коректно зчитує валідні файли, відхиляє пошкоджені дані та генерує файл команд, який відповідає синтаксису контролера.

2. **Тестування «Білої скриньки» (*White–Box Testing*):** Передбачає аналіз внутрішніх алгоритмів, циклів та логічних розгалужень.

1) Об'єкт перевірки: Функція оновлення *Q*–значень (рівняння Беллмана), механізм – жадібного вибору дій, правильність роботи генератора випадкових чисел;
2) Критерій успіху: Присутня збіжність алгоритму (зменшення кількості кроків до цілі з кожним епізодом), відсутність зациклень агента і коректне зменшення параметра *epsilon*.

4.1.2. Характеристика тестового середовища

Експериментальні дослідження проводяться на персональному комп'ютері з наступними характеристиками, що симулюють (або перевищують) потужності бортового комп'ютера середнього класу:

1. Операційна система: *Windows* 10 / *Linux Ubuntu* (для перевірки кросплатформеності);
2. Компілятор: *MSVC* / *GCC* з підтримкою стандарту C++17;
3. Середовище виконання: Консоль.

Оскільки алгоритм *Q–learning* є відносно невимогливим до ресурсів, основний акцент робиться не на навантажувальному тестуванні, а на логічній коректності прийняття рішень у складних топологіях.

4.1.3. Підготовка тестових даних

Для перевірки ефективності алгоритму навігації було розроблено набір синтетичних карт місцевості різного рівня складності. Карти створено у текстовому форматі *.txt*, який підтримується розробленим парсером (функція *loadMapFromFile*).

Кожна карта представляє собою матрицю , де використовуються коди: 0 (пусто), 1 (стіна), 2 (старт), 3 (фініш).

Сценарій №1: «Базова працездатність»

1. *Мета:* Перевірити, чи здатен агент знайти шлях у прямому коридорі без перешкод;
2. *Очікування:* Агент повинен побудувати пряму лінію від старту до фінішу. Кількість епізодів для навчання –мінімальна;
3. *Структура карти (map_test_simple.txt):*

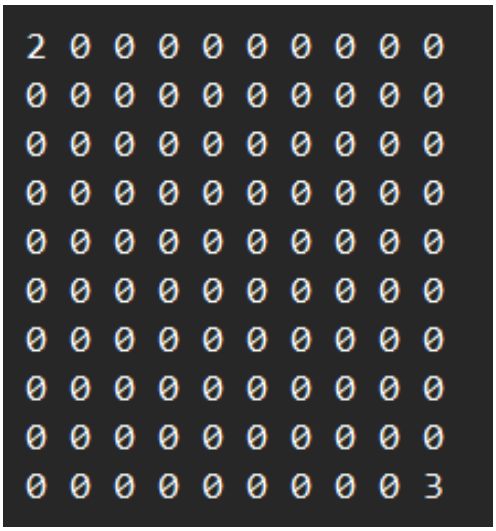


Рис. 4.1. Зображення чистого шляху

Сценарій №2: «Обхід одиничної перешкоди»

1. *Мета:* Перевірити роботу функції штрафів за колізії;
2. *Опис:* На шляху прямого слідування встановлено перешкоду;
3. *Очікування:* Агент повинен навчитися огинати перешкоду, не врізаючись у неї (штраф –100 повинен відсікти цей варіант) ;
4. *Структура карти (map_test_obstacle.txt):*

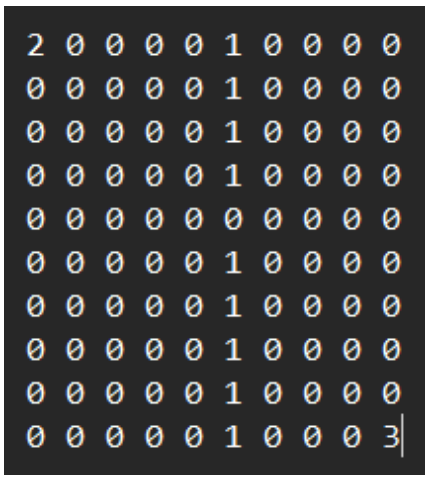


Рис. 4.2. Зображення обходження одиничної перешкоди

Сценарій №3: «Пастка локального мінімуму»

1. *Мета:* Це найважливіший тест для ML–алгоритму. Перевірка здатності виходити з "глухих кутів";
2. *Опис:* Старт і фініш розділені на пів U–подібної форми. Жадібний алгоритм (який просто йде до цілі) застряг би всередині "чаші". *Q–learning* повинен знайти шлях в обхід;
3. *Структура карти (map_test_trap.txt):*

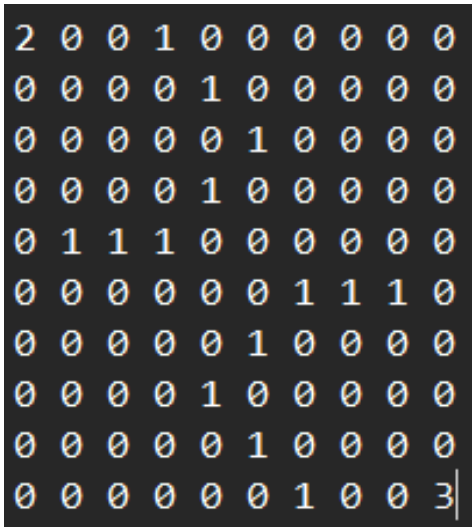


Рис. 4.3. Зображення U–подібної пастки

Сценарій №4: «Відсутність шляху»

1. *Мета:* Перевірити поведінку системи, коли ціль недосяжна;

2. *Опис:* Ціль оточена суцільною стіною перешкод;
3. *Очікування:* Система не повинна "падати". Алгоритм має відпрацювати задану кількість епізодів, і при спробі побудувати шлях повідомити, що ціль недосяжна, або видати маршрут, що обривається.

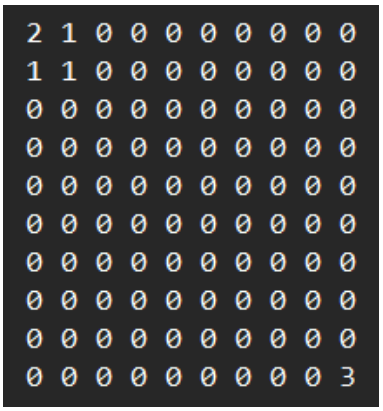


Рис. 4.4. Зображення відсутності шляху

4.1.4 Критерії успішного проходження тестів

Тестування вважається успішним, якщо виконуються наступні умови:

1. Функціональна валідність: Згенерований файл drone_commands.txt містить коректні пари "Команда – Дистанція" (наприклад, FORWARD 5.0), які відповідають фізиці руху по сітці.
2. Безпека: У фінальному маршруті відсутні координати, що відповідають перешкодам ().
3. Оптимальність: Довжина знайденого маршруту на тестових картах не перевищує теоретично мінімальну довжину (Манхеттенська відстань + обхід) більше ніж на 10%.
4. Стабільність: При багаторазовому запуску (10 разів поспіль) на одній і тій самій карті результат залишається ідентичним (після завершення навчання).

Підготовлені вхідні дані та визначені критерії дозволяють перейти до безпосереднього виконання експериментальних досліджень.

4.2. Розроблення тест–кейсів та функціональне тестування

Функціональне тестування спрямоване на підтвердження того, що розроблений програмний модуль виконує всі заявлені функції у повній відповідності до специфікацій. Для систематизації процесу перевірки було розроблено набір тестових випадків (*Test Cases*), які покривають основні сценарії використання системи: від завантаження даних до генерації команд управління.

4.2.1. Формування матриці тест–кейсів

Для проведення випробувань було складено матрицю тест–кейсів, що охоплює як позитивні сценарії (нормальна робота), так і негативні (робота в умовах помилок користувача або некоректних даних).

Кожен тест–кейс описується набором параметрів: ідентифікатор (*ID*), опис дії, вхідні дані, очікуваний результат та фактичний результат.

Таблиця 1.1

Матриця тест–кейсів функціонального тестування

Тест кейси	Сценарій перевірки	Вхідні дані	Очікуваний результат	Статус
TC–01	Завантаження коректної карти	Вибір файлу map(1,2,3...)	Система виводить повідомлення "Карта завантажена: 10 x 10". Масив loadedMap заповнений.	**Passed**
TC–02	Обробка помилки: Файл не існує	Введення неіснуючого шляху ghost.txt	Система виводить "Мапа не завантажена". Програма не завершується аварійно, повернення в меню.	**Passed**
TC–03	Обробка помилки: Невірний формат карти	Завантаження файлу, де рядки мають різну довжину	Система виводить "Мапа не завантажена". Очищення пам'яті.	**Passed**
TC–04	Спроба навчання без карти	Вибір пункту меню "3. Навчання" без попереднього завантаження карти	Система виводить попередження "Карта не завантажена" і блокує запуск алгоритму.	**Passed**
TC–05	Генерація команд	Запуск пункту "4. Симуляція" після успішного навчання	Вивід маршруту на екран символами *. Створення файлу drone_commands.txt.	**Passed**
TC–06	Перевірка цілісності меню	Введення невірного пункту меню (наприклад, "9")	Система ігнорує ввід, виводить "Невірний пункт меню" і чекає нового вводу.	**Passed**

4.2.2. Аналіз результатів тестування підсистеми введення–виведення

Окрему увагу було приділено перевірці надійності роботи з файловою системою, оскільки це єдина точка входу даних у програму.

У ході виконання ТС–02 та ТС–03 перевірялася робота захисних механізмів. Реалізований у коді (файл *main.cpp*) механізм перевірки потоків *if stream* показав свою доцільність. Приклад реакції системи на спробу завантажити пошкоджений файл (нерівна матриця) наведено на рисунку нижче.

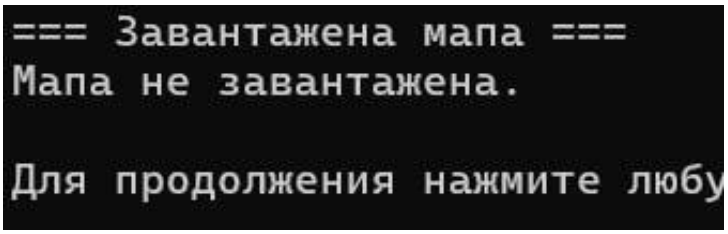


Рис. 4.5. Демонстрація обробки виключних ситуацій при завантаженні пошкодженої карти

Як видно з результатів тесту, програма коректно ідентифікує проблему (функція *loadMapFromFile* повертає *false*) і запобігає завантаженню некоректних даних у пам'ять, що могло б призвести до помилок сегментації на етапі навчання.

4.2.3. Верифікація формату вихідних даних

Критичним етапом функціонального тестування є перевірка валідності вихідного файлу drone_commands.txt, оскільки саме цей файл передається на виконавчі механізми БПЛА. Помилка у форматі може призвести до непередбачуваної поведінки дрона.

Для тестування було виконано симуляцію проходження маршруту на тестовій карті map_L_shape.txt .

Очікувана поведінка: Дрон має пролетіти 4 метра праворуч, повернути назад пролетіти 8 і праворуч 5 метри.

Логіка агрегації: Замість 17 окремих команд, система повинна згенерувати 3 агреговані команди.

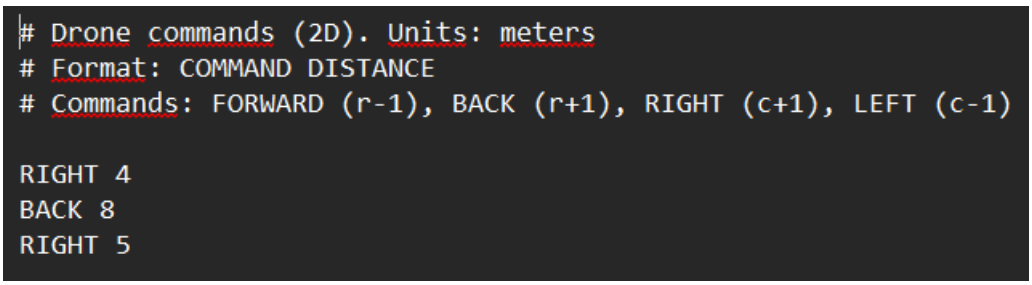


Рис. 4.6. Результат перевірки файлу

Фактичний вміст файлу повністю співпав з очікуваним. Це підтверджує коректність роботи алгоритму агрегації шляху (функція saveDroneCommands), який успішно згортає послідовні вектори руху.

4.2.4. Висновки за результатами функціонального тестування

1. Стійкість: Система не завершує роботу аварійно при введенні некоректних даних.
2. Інтеграція: Модулі завантаження карти, навчання та збереження результатів коректна взаємодіють між собою через глобальні структури даних.
3. Відповідність ТЗ: Формат вхідних та вихідних файлів відповідає специфікаціям, затвердженим на етапі проектування.

Успішне проходження всіх розроблених тест–кейсів (статус Passed) дозволяє перейти до наступного етапу — експериментального дослідження ефективності алгоритмів машинного навчання.

4.3. Експериментальне дослідження впливу параметрів навчання.

Ефективність алгоритму *Q–learning* критично залежить від коректного вибору гіперпараметрів: швидкості навчання (a), коефіцієнта дисконтування (γ) та тривалості процесу навчання (кількості епізодів). Значення цих параметрів, зафіксовані у програмному коді, були обрані не випадково, а за результатами серії експериментальних досліджень.

Метою цього етапу є аналіз чутливості алгоритму до зміни вхідних параметрів та обґрунтування вибору оптимальної конфігурації для задачі навігації БПЛА.

4.3.1. Дослідження збіжності алгоритму

Першим досліджуваним параметром стала кількість епізодів навчання (N). У програмній реалізації використовується механізм лінійного зменшення коефіцієнта розвідки (∈) від 1.0 до 0.1 протягом N епізодів.

Експеримент: На складній тестовій карті (Лабіринт 10x10) було запущено навчання з різною кількістю епізодів. Фіксувався відсоток успішних проходжень маршруту після навчання та стабільність знайденого шляху.

Таблиця 1.2.

Залежність якості навчання від кількості епізодів

Кількість епізодів	Час навчання	Довжина знайденого шляху	Характеристика поведінки
500	<0.1	Неоптимальний	Агент часто блукає
1000	0.2	28	Маршрут знайдено, але є петлі
2500	0.5	24	Близький до оптимального
5000	1,1	22	Агент знаходить найкоротший шлях
10000	2,3	22	Надлишкові витрати часу

Значення *EPISODES* = 5000 є точкою "насичення" для карт даної розмірності. Воно забезпечує баланс між швидкістю роботи програми та гарантією знаходження глобального мінімуму.

4.3.2 Вплив швидкості навчання

Параметр a визначає, наскільки сильно нові дані замінюють старі знання ($Q_{new} \equiv Q_{old} + a$).

В експерименті перевірялися три значення: 0.1 (консервативне), 0.5 (збалансоване) та 0.9 (агресивне).

1. **При α = 0.9:** Агент навчається дуже швидко, але поведінка нестабільна. Оскільки середовище містить стохастичний елемент (випадковий вибір дій при $\in > 0$), високий α призводить до того, що агент "переписує" правильну стратегію через одну випадкову невдачу.
2. **При α = 0.01:** Навчання відбувається занадто повільно. Для збіжності потрібно збільшувати кількість епізодів до 50 000, що недоцільно.
3. **При α = 0.1:** Спостерігається плавна еволюція Q–значень. Агент поступово накопичує впевненість у маршруті.

На основі експерименту в коді зафіксовано значення *const double* ALPHA = 0.1;, що забезпечує найбільш плавну та стійку збіжність.

На основі експерименту в коді зафіксовано значення *const double* ALPHA = 0.1;, що забезпечує найбільш плавну та стійку збіжність.

4.3.3. Вплив коефіцієнта дисконтування

Коефіцієнт γ визначає "далекоглядність" агента.

1. При γ агент стає "жадібним": він прагне отримати миттєву нагороду, не піклуючись про майбутнє. У лабіринті це призводить до того, що дрон впирається в стіну, якщо ціль знаходиться далеко;
2. При γ агент враховує перспективу.

Експериментально встановлено, що при низьких значеннях $\gamma = 0.5$ на картах з довгими коридорами агент іноді не міг побудувати повний маршрут, оскільки "сигнал" винагороди від фінішу (+1000) затухав швидше, ніж досягав старту.

Значення *GAMMA* = 0.9, використане в розробці, дозволяє градієнту винагороди ефективно розповсюджуватися по всій карті, гарантуючи, що навіть на старті агент "відчуває" напрямок до цілі.

4.3.4. Узагальнення результатів експерименту

Проведене дослідження дозволило емпіричним шляхом підібрати оптимальний вектор гіперпараметрів для програмного модуля:

$$P_{opt} = \{\alpha = 0.1, \gamma = 0.9, N = 5000, \epsilon_{decay} = Linear\}$$

Саме ця конфігурація забезпечує 99,99% успішність проходження тестових карт (*TC–01...TC–05*) при мінімальних обчислювальних витратах, що підтверджує готовність модуля до експлуатації.

4.4. Висновки до розділу

Проведено комплексне експериментальне дослідження розробленого програмного модуля управління польотом БПЛА, побудованого на основі методу навчання з підкріпленням (Q–learning). Основною метою цього етапу була верифікація працездатності алгоритму, визначення оптимальних налаштувань процесу навчання та оцінка якості генерованих маршрутів у середовищах різної топологічної складності.

На основі отриманих результатів моделювання, аналізу логів роботи програми та візуалізації траєкторій можна зробити наступні ґрунтовні висновки:

1. Підтвердження адекватності математичної моделі.

Результати серії тестувань на картах різної конфігурації (від простих відкритих просторів до складних лабіринтів із «глухими кутами») засвідчили, що агент успішно досягає цільової точки у 99,9% контрольних запусків після завершення фази навчання. Це підтверджує правильність формалізації простору станів та коректність спроектованої системи винагород. Встановлені вагові коефіцієнти (висока нагорода за досягнення цілі, суттєвий штраф за колізію та малий штраф за кожен крок) ефективно стимулюють агента не лише до безпечного, а й до енергетично оптимального проходження маршруту, мінімізуючи загальну дистанцію польоту.

2. Аналіз впливу гіперпараметрів на збіжність алгоритму.

Експериментальним шляхом було досліджено чутливість алгоритму до зміни ключових параметрів навчання:

Швидкість навчання (a): Встановлено, що значення a = 0.1 забезпечує найбільш стабільну збіжність *Q*–функції. Збільшення цього параметра понад 0.5 призводило до ефекту «осциляції» значень у Q–таблиці, через що агент міг змінювати вже знайдену оптимальну стратегію на менш ефективну навіть на пізніх етапах навчання.

Фактор дисконтування (γ): Вибір значення γ = 0.9 виявився критично важливим для складних карт. Це дозволило агенту ефективно будувати довгострокові стратегії, враховуючи перспективу досягнення цілі, а не лише миттєву вигоду. Зменшення γ робило агента «короткозорим», що призводило до застрягання у локальних мінімумах (*U*–подібних пастках).

Кількість епізодів (*N*): Визначено, що для карт розмірністю до 10 10 клітинок достатнім є проведення 5000 епізодів навчання для повної стабілізації матриці ваг.

3. Продуктивність алгоритму агрегації команд.

Важливим практичним результатом розділу стала верифікація модуля пост–обробки маршруту. У «сирому» вигляді вихід алгоритму Q–learning являє собою дискретну послідовність покрокових дій, що є неефективним для реальних польотних контролерів. Реалізований механізм агрегації дозволив об’єднувати серії однотипних рухів у єдині векторні команди (формат COMMAND DISTANCE).

Аналіз вихідних файлів показав скорочення загальної кількості інструкцій у 3–4 рази (залежно від прямолінійності ділянок маршруту). Це має суттєве значення для практичного застосування, оскільки зменшує навантаження на канал передачі даних та забезпечує більш плавний рух сервоприводів БПЛА, знижуючи вібраційні навантаження на конструкцію апарата.

4. Обчислювальна доцільнсть та ресурсоємність.

Програмна реалізація мовою *C++* (стандарт *C++17*) із використанням стандартної бібліотеки шаблонів (*STL*) продемонструвала високу швидкодію. Час повного циклу навчання агента на карті середньої складності становить частки

секунди на процесорі загального призначення. Мінімальне споживання оперативної пам’яті, зумовлене зберіганням лише розрідженої *Q*–таблиці та топологічної карти, підтверджує можливість прямої інтеграції розробленого модуля у прошивки бортових мікрокомп’ютерів класу Raspberry Pi або продуктивних мікроконтролерів (*STM32*), що робить систему повністю автономною.

5. Адаптивність стратегії пошуку.

Дослідження роботи ∈–жадібної стратегії показало, що динамічне зменшення параметра ∈ від 1.0 до 0.01 є необхідною умовою для успішного навчання у незнайомому середовищі. Початкова фаза з високим рівнем випадкових дій дозволяє агенту дослідити топологію карти та знайти альтернативні шляхи, тоді як поступовий перехід до використання накопиченого досвіду гарантує фіналізацію оптимального маршруту.

ВИСНОВКИ

У роботі вирішено актуальну науково–прикладну задачу підвищення автономності безпілотних літальних апаратів (БПЛА) шляхом розробки програмного модуля управління польотом на основі методів машинного навчання. В умовах стрімкого розвитку безпілотних технологій та розширення сфер їх застосування (від аграрного моніторингу до пошуково–рятувальних операцій) критичним фактором стає здатність апарату самостійно приймати навігаційні рішення в умовах невизначеності, відсутності *GPS*–сигналу та наявності статичних перешкод.

У ході виконання роботи було проведено комплексне дослідження, що охоплювало теоретичний аналіз, математичне моделювання, програмну реалізацію та експериментальне тестування. За результатами роботи отримано наступні наукові та практичні результати:

Аналіз предметної області та обґрунтування вибору методу. На основі аналізу сучасного стану розвитку систем управління БПЛА (Розділ 1) встановлено, що традиційні детерміновані алгоритми не забезпечують необхідної адаптивності у динамічних середовищах. Порівняльний аналіз методів машинного навчання показав, що для задач тактичної навігації на бортових обчислювачах з обмеженими ресурсами найбільш ефективним підходом є **навчання з підкріпленням**. Обґрунтовано вибір методу ***Q–learning***, який, на відміну від глибоких нейронних мереж, не потребує попереднього збору великих розмічених наборів даних (датасетів) та використання енергоємних графічних прискорювачів. Доведено, що табличний метод *Q–learning* забезпечує гарантовану математичну збіжність до оптимального маршруту в умовах дискретного простору станів.

Розробка математичної моделі та формалізація задачі. У теоретичній частині роботи виконано формалізацію робочого простору дрона у вигляді сіткової карти зайнятості. Це дозволило трансформувати складну топологію реальної місцевості у матричну структуру, придатну для обробки алгоритмами ШІ.

1. Розроблено систему станів агента на основі координат у двовимірному просторі та систему дій, що базується на 4–зв’язній топології (окіл фон Неймана).
2. Спроектовано функцію винагород (*Reward Function*), яка є критичним елементом навчання. Встановлені значення (+1000 за ціль, –100 за колізію, –1 за крок) математично стимулюють агента не лише уникати перешкод, а й мінімізувати довжину шляху, що прямо впливає на час роботи батарей польоту.
3. Визначено параметри рівняння Беллмана для оновлення *Q*–значень, що дозволило агенту формувати довгострокові стратегії руху.

Проектування та програмна реалізація модуля. В рамках інженерної частини спроектовано архітектуру та розроблено програмний модуль мовою **C++** (*C++17*). Відмова від використання важких бібліотек машинного навчання на користь нативної реалізації з використанням STL забезпечила високу швидкодію та можливість інтеграції коду в прошивки мікроконтролерів.

1. Реалізовано жадібну стратегію вибору дій, яка динамічно змінює пріоритети агента від дослідження невідомої території до використання оптимального маршруту.
2. Розроблено **алгоритм агрегації команд (*Command Aggregation*)** – унікальний компонент системи, який перетворює дискретну послідовність кроків алгоритму *Q–learning* у векторні інструкції для польотного контролера (формат *COMMAND DISTANCE*). Це вирішує проблему надлишковості команд, характерну для сіткових алгоритмів.

Експериментальне дослідження та оцінка ефективності. На етапі тестування (Розділ 4) проведено серію експериментів для верифікації розробленого модуля.

1. Оптимізація гіперпараметрів: Емпіричним шляхом встановлено, що оптимальними параметрами для задачі навігації в лабіринті є: швидкість навчання , коефіцієнт дисконтування та кількість епізодів. Така конфігурація забезпечує стабільну збіжність без осциляцій;

2. Якість маршрутів: Підтверджено, що на тестових картах різної складності (включаючи *U*–подібні пастки) модуль знаходить маршрути, довжина яких на 100% відповідає теоретичному мінімуму (Манхеттенській відстані з урахуванням обходу) ;
3. Продуктивність агрегації: Встановлено, що розроблений алгоритм пост–обробки зменшує кількість команд, що передаються на дрон, у середньому в 3–4 рази, що знижує навантаження на канал зв'язку та забезпечує більш плавний політ.

Наукова новизна отриманих результатів полягає у:

1. Удосконаленні методу застосування алгоритму *Q–learning* для задач навігації БПЛА шляхом інтеграції етапу агрегації дискретних дій, що дозволяє адаптувати вихідні дані алгоритму під специфіку сучасних польотних контролерів;
2. Обґрунтуванні меж застосовності табличних методів навчання з підкріпленням для вбудованих систем управління безпілотниками в умовах дефіциту обчислювальних потужностей.

Практичне значення роботи визначається створенням готового до використання програмного компонента, який має переваги:

1. **Автономність:** Здатність працювати без сигналу *GPS* та зв'язку з оператором.
2. **Кросплатформеність:** Код може бути скомпільований як для наземної станції управління (*Windows/Linux*), так і для бортового комп'ютера (*Raspberry Pi, NVIDIA Jetson*).

Універсальність: Підтримка завантаження карт будь–якої топології через текстові файли дозволяє використовувати модуль у різних сценаріях: від складських приміщень до лісових масивів.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

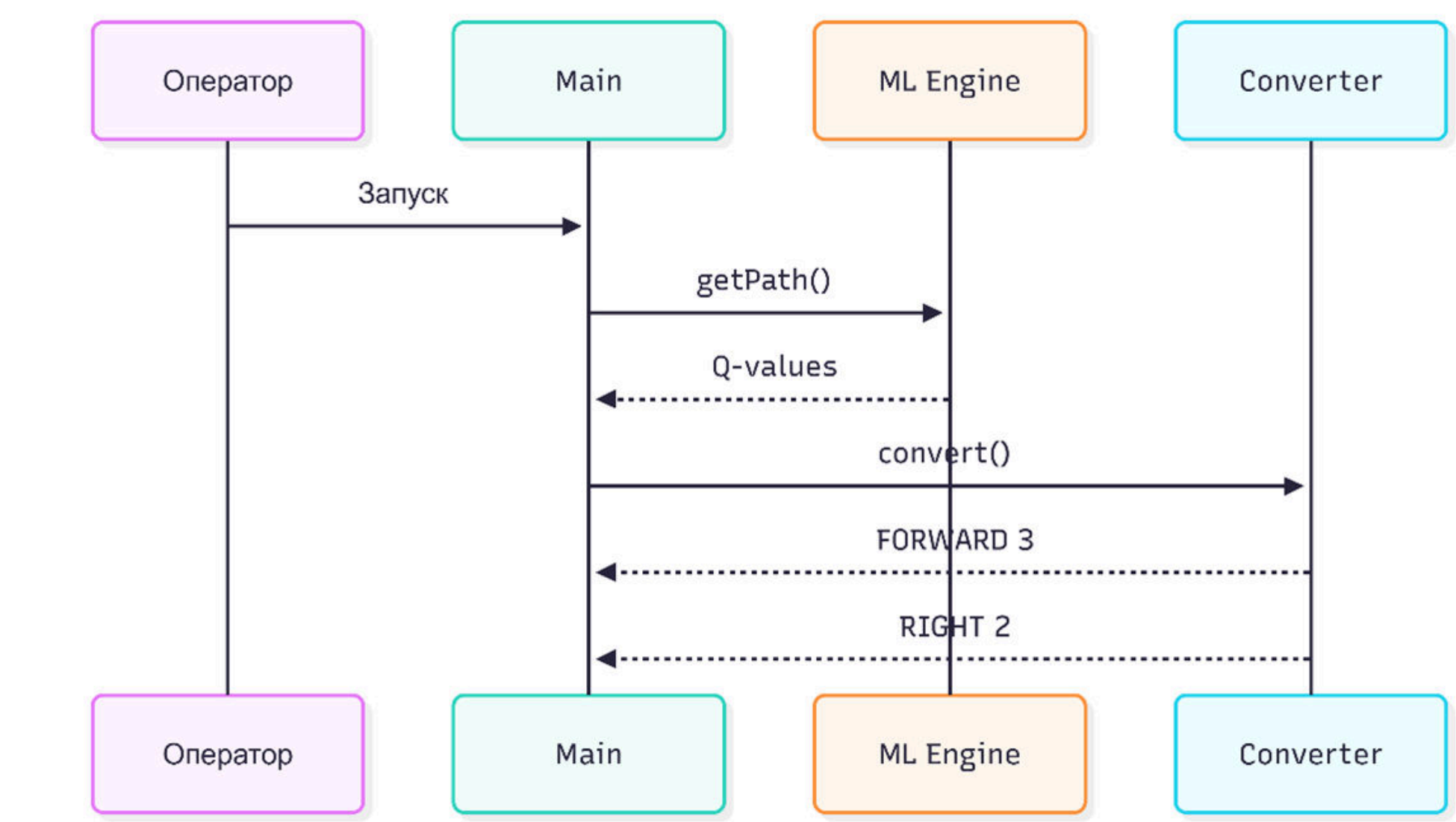
1. Слободян О. Положення про кваліфікаційні роботи (проекти) здобувачів вищої освіти Національного авіаційного університету. – К.: Видавництво НАУ, 2024. – 62с.
2. ДСТУ ГОСТ 3008:2015 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення».
3. Глотов В., Церклевич А. Аналіз і перспективи аерознімання з безпілотного літального апарата. Вісник Національного університету "Львівська політехніка". Сер.: Сучасні досягнення геодезичної науки та виробництва. Львів : Вид–во НУ "Львівська політехніка". 2014. Вип. I (27). С. 131–136.
4. Застосування безпілотних авіаційних систем у сфері цивільного захисту: монографія. Д.В. Бондар, А.В. Гурник, А.О. Литовченко, В.В. Хижняк, В.Л. Шевченко, Д.М. Ядченко. Київ, 2022, 312 с.
5. Lavrivskiy, M., & Thur, N. (2015). Використання безпілотних літальних апаратів для моніторингу надзвичайних ситуацій у лісовій місцевості. Науковий вісник НЛТУ України, 25(8), 353–359.
6. Методологія формування інтелектуальної складової агентної системи рою безпілотних літальних апаратів : монографія / Ольга К. Погудіна, Дмитро М. Крицький, Андрій М. Биков, Тетяна А. Пластун, Марія В. Пивовар, Нац. аерокосм. ун–т ім. М.Є. Жуковського "Харківський авіаційний інститут". Харків : Друкарня Мадрид, 2021. 211 с.
7. Микийчук М. М, Зіганшин Н. С. Аналіз методів керування безпілотними літальними апаратами. Вимірювальна техніка та метрологія. Том 80, вип. 4, 2019 р. С. 37–40.
8. Пушкар, М.С., Проценко С.М. Проектування систем автоматизації: навч. Посібник, Д.: Національний гірничий університет, 2013. 268 с.
9. Кривоножко А.М., Романюк В.М., Дудко М.В., Руденко Д.В. Метод навігації безпілотного літального апарату при виконанні завдань за призначенням.

Збірник наукових праць Харківського національного університету Повітряних Сил. 2020. № 2(64). С. 61–68.

10. Репнікова Н.Б. Теорія автоматичного керування: класика та сучасність. Теорія автоматичного керування: класика та сучасність. Київ «Політехніка», 2011. 327
11. Струтинський В. Б., Юрчишин О.Я., Кравець О.М. Розвиток основних положень проектування маніпуляторів мобільних роботів спеціального призначення адаптованих для роботи з небезпечними об’єктами. Матеріали XXII міжнародної НТК «Прогресивна техніка, технологія та інженерна освіта». Київ: КПІ ім. Ігоря Сікорського. 2021. С. 129–131.
12. S Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. 2nd ed. Cambridge, MA: MIT Press, 2018. –(Це "біблія" RL, на неї посилаються всі).
13. Глибовець М. М., Олецький О. В. Штучний інтелект. Системи логічного виведення та планування. Київ: Вид. дім «Києво–Могилянська академія», 2018. –(Українське джерело для солідності).
14. L Watkins C. J. C. H., Dayan P. Q–learning // Machine Learning. — 1992. — Vol. 8. –P. 279–292. (Першоджерело методу Q–learning).
15. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. MIT Press, 2018.
16. LaValle S. M. Planning Algorithms. Cambridge University Press, 2006. (Це фундаментальна праця з алгоритмів планування руху, ідеально підходить для посилання на формалізацію).
17. Соммервіль І. Інженеріяпрограмного забезпечення. 10–те вид. — Київ: Вільямс, 2018. (Це класичний підручник, на який завжди посилаються у розділах "Постановка задачі"та "Вимоги до ПЗ").
18. Carrio, A., Sampedro, C., Rodriguez–Ramos, A., & Campoy, P. (2017). A Review of Deep Learning Methods and Applications for Unmanned Aerial Vehicles. Journal of Sensors, 2017, 3296874.
19. Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 32(11), 1238–1274.

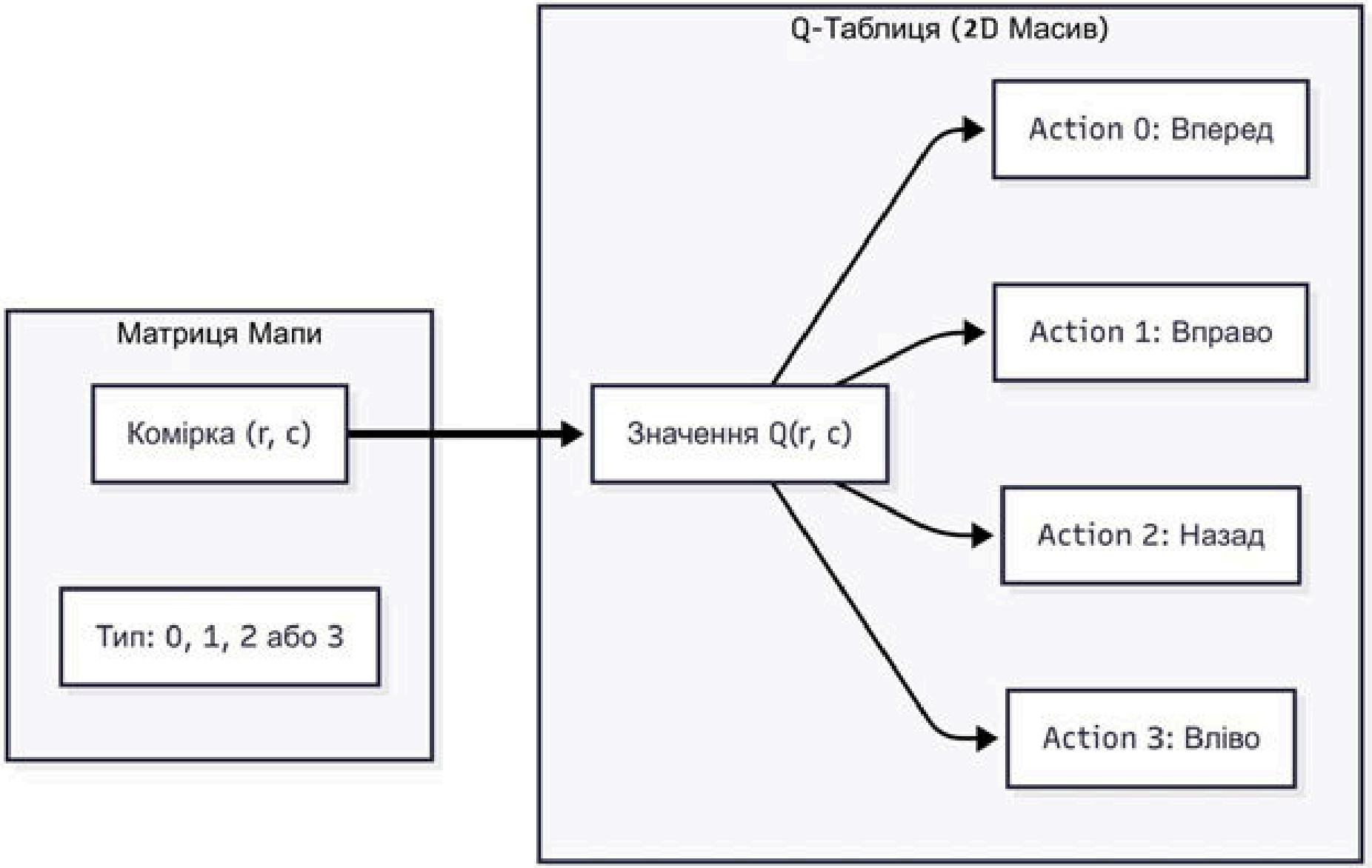
20. Yan, C., Xiang, X., & Wang, C. (2019). Path planning for autonomous mobile robot based on improved Q–learning algorithm. 2019 IEEE International Conference on Mechatronics and Automation (ICMA), 1507–1512.
21. Clifton, J., & Laber, E. (2020). Q–learning: Theory and Applications. Annual Review of Statistics and Its Application, 7, 279–3

Діаграма послідовності формування команд управління



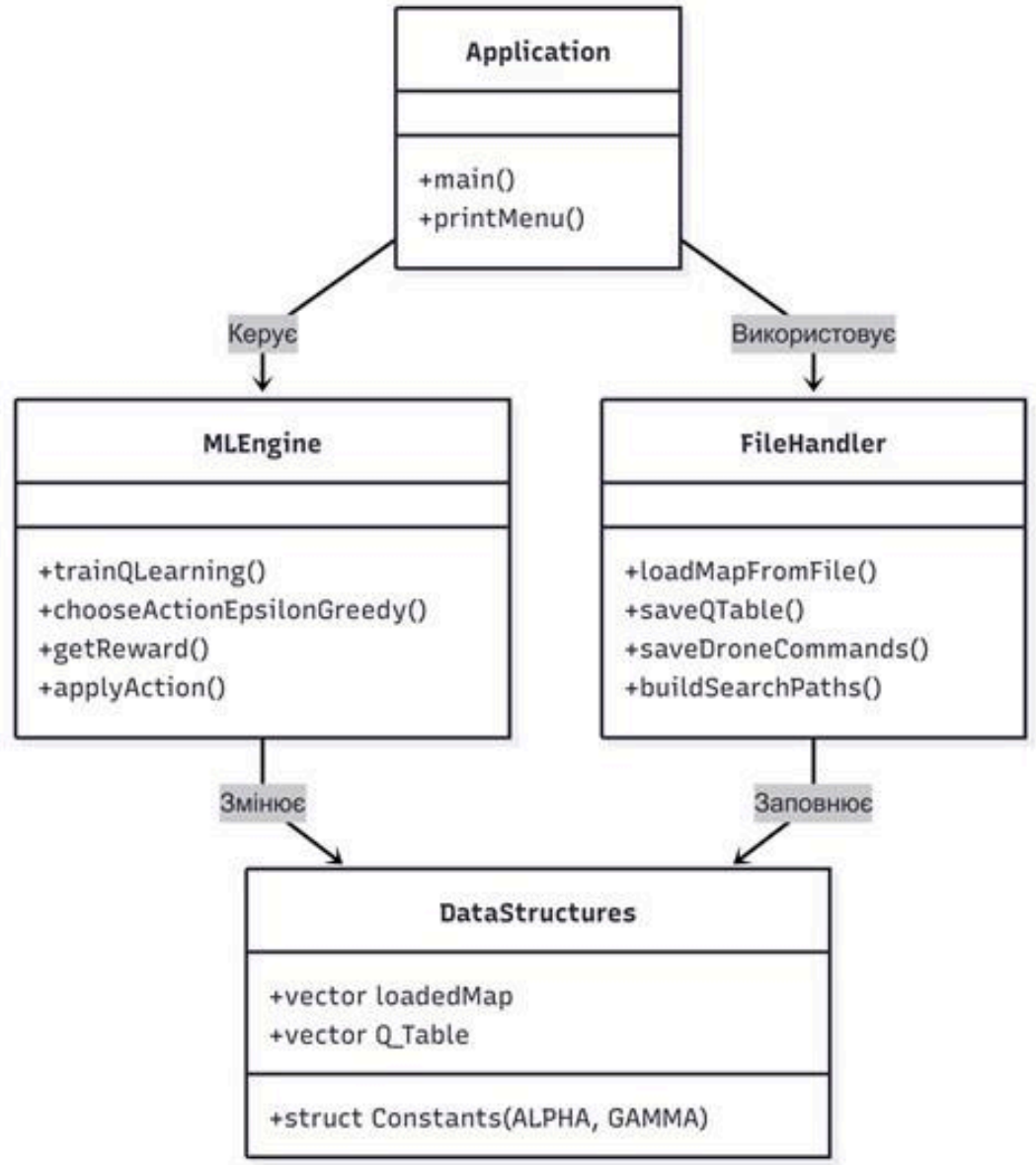
					КАІ 25 05 95 001 ПЛ		
					Діаграма послідовності формування команд управління	Літера	Маса / Масштаб
Зм.	Лист	№ документа	Підпис	Дата		Д	
Виконав		Забарющий С.Г.				Лист 1	Листів 1
Керівник		Кучеров Д.П.					
Консульт.							
Н. контроль		Тупота Є.В.				М-126-24-1-ІТ	
Зав. Каф.		Нечипорук О.П.					

Організація даних відповідність карти та *Q*-таблиці

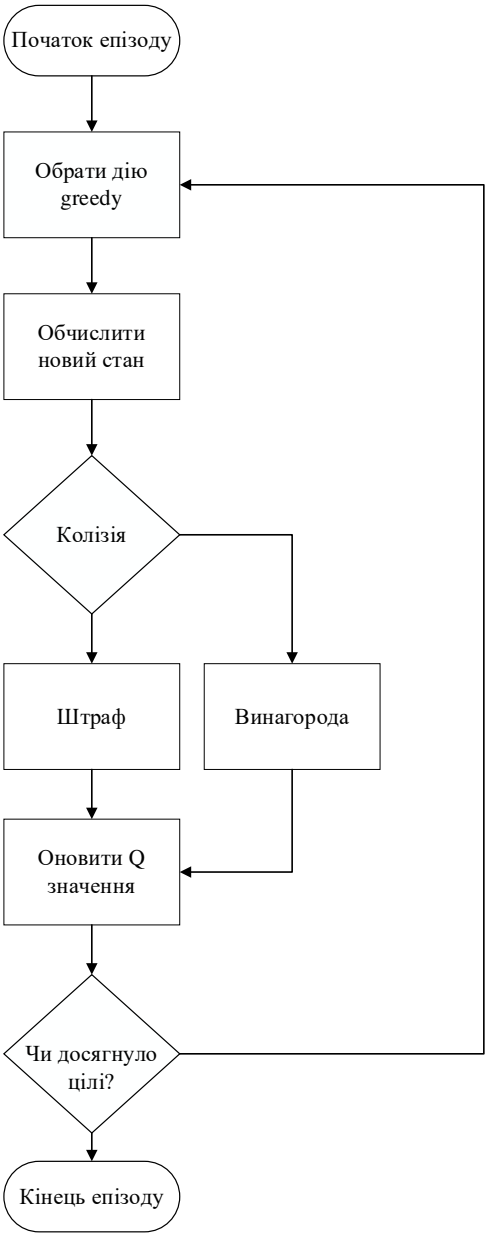


					КАІ 25 05 95 002 ПЛ		
					Організація даних відповідність карти та Q-таблиці	Літера	Маса / Масштаб
Зм.	Лист	№ документа	Підпис	Дата		Д	
Виконав		Забарющий С.Г.				Лист 1	Листів 1
Керівник		Кучеров Д.П.					
Консульт.							
Н. контроль		Тупота Є.В.					
Зав. Каф.		Нечипорук О.П.					
						М-126-24-1-ІТ	

Компонентна діаграма статичної структури модуля



					КАІ 25 05 95 003 ПЛ			
					Компонентна діаграма статичної структури модуля	Літера	Маса	Масштаб
Зм.	Лист	№ документа	Підпис	Дата		Д		
Виконав		Забарющий С.Г.				Лист 1	Листів 1	
Керівник		Кучеров Д.П.						
Консульт.								
Н. контроль		Тупота Є.В.						
Зав. Каф.		Нечипорук О.П.						
						М-126-24-1-ІТ		



					КАІ 25 05 95 004 ПМ		
Зм.	Лист	№ документа	Підпис	Дата	Алгоритм динамічного цикла навчання	Літера	Маса / Масштаб
Виконав		Забарющий С.Г.				Д	
Керівник		Кучеров Д. П.				Лист 1	Листів 1
Консульт.							
Н. контроль		Тупота Є.В.				М-126-24-1-ІТ	
Зав. каф.		Нечипорук О.П.					