Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» червня 2025 р.

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «**Системи автоматизації керування рухомими об'єктами**»

Спеціальність 123 – Комп'ютерна інженерія.
Галузь знань: 12 – Інформаційні технології.
Освітня програма «Комп'ютерна інженерія».

Захищено на засіданні
Екзаменаційної комісії № 44
протокол № __ від __.06.2025 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **ЧУГАЙ А.М.**

Виконав:
Студент групи КІ– 41
Власов Михайло Андрійович

Керівник: д.т.н., проф. ЗВО кафедри комп'ютерних систем та робототехніки
Мірошник М. А.

Рецензент: д.т.н., проф. ЗВО кафедри Інформаційних технологій УкрДУЗТ
ДОЦЕНКО С. І.

Харків – 2025

АНОТАЦІЯ

Обсяг пояснювальної записки бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і двох додатків. Загальний обсяг роботи складає 61 сторінки, із яких 49 сторінок основної частини з 24 рисунками, 8 таблицями, 3 формулами, 20 найменуваннями списку використаних джерел та двома додатками.

Метою роботи є розробка, моделювання та експериментальна перевірка автономної системи керування дроном на основі алгоритму A* у середовищі Webots з інтеграцією стабілізаційного PID-регулятора, системи уникнення перешкод та планувальника маршруту в структурі керування на основі автомата станів.

Об’єкт дослідження – процеси автономного керування рухомими об'єктами в динамічному середовищі.

Предмет дослідження – архітектура та алгоритми керування дроном з використанням класичних та евристичних методів у симуляційному середовищі.

У результаті проведеного дослідження проаналізовано класифікацію та архітектуру автоматизованих систем керування рухомими об’єктами, досліджено технічну базу автономного керування, розроблено віртуальну модель квадрокоптера, реалізовано PID-регулятори стабілізації та навігаційний алгоритм A*, інтегрований у FSM-контролер. Проведено тестування системи у сценаріях з перешкодами, що підтвердило її точність ($\sigma \approx 0.9$ м), ефективність планування (1.2 мс) і економію часу місії на 18%.

Результати роботи можуть бути використані для проєктування автономних роботизованих систем, оптимізації траєкторного планування в мобільній робототехніці, розробки безпілотних платформ у сфері моніторингу, доставки, охорони та в наукових дослідженнях.

Ключові слова: автономна навігація, дрон, алгоритм A*, PID-регулятор, автомат станів, уникнення перешкод, Webots.

ABSTRACT

The explanatory note to the bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references, and two appendices. The total volume of the work is 61 pages, including 49 pages of the main part, with 24 figures, 8 tables, 3 formulas, 20 references, and two appendices.

The aim of the work is to design, simulate, and experimentally validate an autonomous drone control system based on the A* algorithm in the Webots environment, integrating a PID stabilization controller, obstacle avoidance system, and path planner within a finite state machine (FSM) architecture.

The object of research is the processes of autonomous control of moving objects in dynamic environments.

The subject of research is the architecture and algorithms for drone control using classical and heuristic methods in a virtual simulation environment.

As a result of the study, a comprehensive analysis of the classification and architecture of automated control systems for mobile objects was conducted. The technical basis of autonomous control was explored, a virtual drone model was developed, PID controllers for stabilization and an A*-based navigation algorithm were implemented and integrated into an FSM controller. System testing in obstacle-rich scenarios confirmed high accuracy ($\sigma \approx 0.9$ m), fast path planning (1.2 ms), and a mission time reduction of 18%.

The results of this work can be applied to the development of autonomous robotic systems, trajectory planning optimization in mobile robotics, and the design of unmanned platforms for monitoring, delivery, surveillance, and research purposes.

Keywords: autonomous navigation, drone, A* algorithm, PID controller, finite state machine, obstacle avoidance, Webots.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 1

ВСТУП 3

РОЗДІЛ 1. ОГЛЯД АВТОМАТИЗОВАНИХ СИСТЕМ КЕРУВАННЯ РУХОМИМИ ОБ'ЄКТАМИ 4

1.1 Класифікація автоматизованих систем керування рухомими об'єктами 5

1.2 Методи та алгоритми керування рухомими об'єктами 8

1.3 Сучасні тенденції та виклики у сфері автономного керування 12

Висновки за розділом 1 15

РОЗДІЛ 2. ТЕХНІЧНІ ОСНОВИ АВТОМАТИЗОВАНИХ СИСТЕМ КЕРУВАННЯ РУХОМИМИ ОБ'ЄКТАМИ 16

2.1 Апаратна архітектура автоматизованих систем керування 17

2.2 Програмна архітектура автоматизованих систем керування 22

2.3 Синергія цифрових двійників та машинного навчання в системах управління 27

Висновки за розділом 2 30

РОЗДІЛ 3. РОЗРОБКА ТА МОДЕЛЮВАННЯ СИСТЕМИ КЕРУВАННЯ ДРОНОМ У СЕРЕДОВИЩІ WEBOTS 32

3.1 Віртуальний прототип дрона 32

3.2 Реалізація системи керування 34

3.3 Удосконалення системи керування 39

3.4 Автономне керування дроном на основі алгоритму A* 41

Висновки за розділом 3 45

ВИСНОВКИ 47

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 49

ДОДАТКИ 52

ПЕРЕЛІК СКОРОЧЕНЬ

CAN	— Controller Area Network
CARLA	—симуляційне середовище для автомобілів
DDS	— Data Distribution Service
DDPG	— Deep Deterministic Policy Gradient
DCS	— Distributed Control System
FSM	— Finite State Machine
GPS	— Global Positioning System
IMU	— Inertial Measurement Unit
IoT	— Internet of Things
LED	— Light-Emitting Diode
MPC	— Model Predictive Control
MQTT	— Message Queuing Telemetry Transport
OPC UA	— Open Platform Communications Unified Architecture
PAC	— Programmable Automation Controller
PID	— Proportional-Integral-Derivative
PLC	— Programmable Logic Controller
RL	— Reinforcement Learning
ROS	— Robot Operating System
SCADA	— Supervisory Control and Data Acquisition
RRT	— Rapidly-exploring Random Tree
RRT*	— Rapidly-exploring Random Tree Star
АСК	— Автоматизована система керування
АСКРО	— Автоматизована система керування рухомими об’єктами
ВМ	— Виконавчий механізм
ДК	— Джерело живлення
ЗЗ	— Зворотний зв’язок

КІ	— Кутова інерція
МН	— Машинне навчання
ОС	— Операційна система
ОСЗП	— Операційна система загального призначення
ОСРЧ	— Операційна система реального часу
ПЛК	— Програмований логічний контролер
СЗЗ	— Система зворотного зв’язку
ЦД	— Цифровий двійник

ВСТУП

Автоматизація керування технічними системами, особливо такими, що перебувають у русі, відкриває нові можливості для підвищення точності, безпеки та ефективності їх функціонування. Зокрема, автономні літальні апарати є прикладом складних об'єктів, які вимагають поєднання обчислювальних алгоритмів, сенсорної обробки й адаптивної логіки для виконання завдань у непередбачуваних умовах.

Автоматизовані системи керування дозволяють реалізовувати повний цикл управління об'єктами: від зчитування інформації з навколишнього середовища до ухвалення рішень і генерації дій на основі алгоритмів стабілізації, навігації та планування. Розробка таких систем є міждисциплінарним завданням, що об'єднує знання з робототехніки, інформатики, кібернетики й математичного моделювання.

Метою цієї роботи є створення та моделювання автономної системи керування квадрокоптером у симуляційному середовищі Webots з використанням алгоритму планування A*, реалізацією стабілізації на основі PID-регуляторів та інтеграцією всіх компонентів у єдиний контролер станів.

Актуальність дослідження зумовлена зростаючим попитом на безпілотні системи у сферах спостереження, картографії, доставки та надзвичайних ситуацій. У таких застосуваннях критичною є здатність апарата самостійно орієнтуватися в середовищі з перешкодами, виконувати завдання без втручання оператора та адаптувати свою поведінку в реальному часі.

Гіпотеза полягає в тому, що поєднання класичних регуляторів (PID) з алгоритмами планування шляху (A*) у рамках автомату станів дозволить забезпечити стабільну, точну та ефективну автономну навігацію дрона у складному середовищі.

РОЗДІЛ 1
ОГЛЯД АВТОМАТИЗОВАНИХ СИСТЕМ КЕРУВАННЯ РУХОМИМИ ОБ'ЄКТАМИ

Автоматизовані системи керування рухомими об'єктами є невід’ємною частиною сучасної техніки та інженерії. Вони знаходять застосування у промисловості, транспорті, авіації, космічних дослідженнях та навіть у побутовій сфері. Основним завданням таких систем є забезпечення автономного або напівавтономного керування об'єктами в реальному часі з використанням сенсорів, обчислювальних алгоритмів і виконавчих механізмів.

Історія розвитку автоматизованих систем керування сягає XVIII століття, коли з’явилися перші механічні регулятори. Одним із перших винаходів у цій сфері став відцентровий регулятор Джеймса Ватта (1788 р.) (рис. 1.1) [1], який дозволяв підтримувати стабільну швидкість парової машини, застосовуючи принцип зворотного зв'язку регулювання параметрів системи.

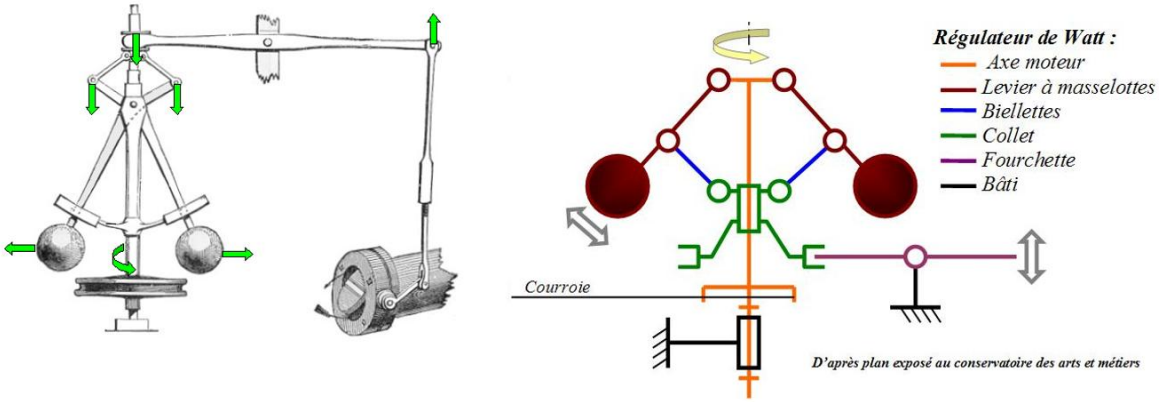

Рисунок 1.1 – Спосіб дії та кінематична схема відцентрованого регулятора [1]

Подальший розвиток автоматизації відбувався завдяки науковим відкриттям та технічним досягненням. У 1912 році Лоуренс Сперрі розробив гіроскопічний автопілот для кораблів, що значно підвищило точність навігації [2]. У 1943 році Норберт Вінер сформулював основи кібернетики, що дало поштовх до розробки сучасних алгоритмів керування.

З появою мікропроцесорів у 1970-1980-х роках автоматизовані системи стали цифровими, що дало потенціал гнучкому програмуванню та інтегруванню

в складніші механізми. Створення програмованих логічних контролерів у 1968 році значно спростило автоматизацію виробничих процесів.

Результати впровадження автоматизованих систем керування в різних сферах є вражаючими, а саме у зниженні впливу людського фактора, збільшення продуктивності та оптимізації витрат.

1.1 Класифікація автоматизованих систем керування рухомими об'єктами

Автоматизовані системи керування рухомими об’єктами (АСКРО) – це комплекс технічних, програмних і організаційних засобів, що забезпечують управління рухомими механізмами за участю оператора [3].

Відповідно до загальноприйнятих визначень, автоматизована система керування (АСК)– це система, яка виконує керуючі дії за участю оператора, з можливістю часткового або повного делегування окремих функцій технічним засобам. Людина виконує функцію контролю, прийняття критичних рішень або коригування роботи системи, тоді як технічні компоненти забезпечують стабільне виконання завдань[4].

Важливим компонентом є система зворотного зв’язку, що дозволяє постійно контролювати вихідні параметри системи, зокрема положення, швидкість та траєкторію руху. Використання датчиків, аналітичних алгоритмів і програмних засобів сприяє швидкій адаптації системи до змін у зовнішньому середовищі, що підвищує її надійність, безпеку і точність роботи.

Крім того, АСК мають здатність коригувати свої параметри відповідно до вхідних даних оператора або змін у навколишньому середовищі. Наприклад, у безпілотних літальних апаратах оператор може задавати траєкторію, а система самостійно адаптує маршрут, враховуючи погодні умови або наявність перешкод.

З позитивних прикладів застосування АСК можна навести випадок з автопілотом у авіації. У 2009 році під час польоту рейсу 1549 авіакомпанії US Airways, після зіткнення з птахами, автопілот допоміг стабілізувати літак, що дозволило екіпажу успішно здійснити аварійну посадку на річку Гудзон без

жертв [5]. Завдяки сучасним автоматизованим системам літак залишався керованим, а злагоджена робота пілотів і технологій врятувала життя всіх пасажирів.

Ще одним яскравим прикладом є системи автоматичного гальмування в автомобілях, найпоширенішими із яких є TESLA [6], запобігаючи зіткненням і рятуючи життя пішоходів та водіїв, та навіть пошкоджень. Такі технології значно зменшили кількість дорожньо-транспортних пригод, особливо в умовах міського трафіку, де людський фактор може призводити до небезпечних ситуацій.

Проте, незважаючи на переваги автоматизованих систем, їх впровадження не завжди проходить без труднощів. Одним із найвідоміших негативних прикладів є аварії літаків Boeing 737 MAX у 2018 та 2019 роках. Через несправність системи MCAS, яка мала допомагати пілотам коригувати кут атаки літака, сталися дві катастрофи, що призвели до загибелі 346 осіб [7]. Проблема полягала в некоректній роботі датчиків, що спричинило помилкове включення системи стабілізації, а відсутність належної можливості швидкого втручання з боку пілотів ускладнила ситуацію.

Ще один приклад небезпечного збою автоматизованих систем стався у 2018 році в Аризоні, США, коли автономний автомобіль Uber наїхав на пішохода, що спричинило смертельний випадок. Розслідування показало, що система не змогла правильно ідентифікувати пішохода та вчасно зреагувати на зміну дорожньої обстановки [8].

Для ефективного аналізу та подальшої роботи в сфері АСК доцільно розглянути основні відмінності, що дозволяють обрати оптимальний підхід до розробки, впровадження та вдосконалення. Класифікувати АСК можна за кількома основними критеріями: за рівнем автономності, сферою застосування, типом керування або архітектурою (рис. 1.2).

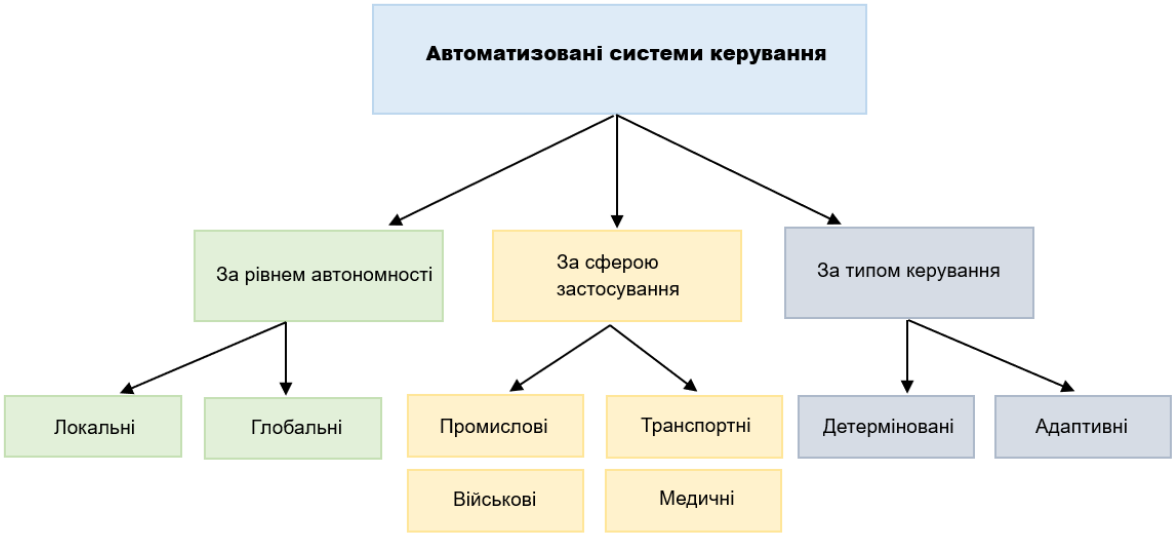

Рисунок 1.2 – Класифікація автоматизованих систем керування

Локальні автоматизовані системи – за рівнем автономності - працюють в межах одного пристрою або вузької системи, тоді як глобальні об'єднують деяку кількість взаємопов’язаних підсистем і керують великими об’єктами. До прикладу можна віднести розумні термостати та централізовану систему управління міським транспортом відповідно.

За сферою застосування поділяють на промислові, транспортні, військові та медичні. Подібні АСК можуть використовуватися на виробництвах, для керування літаками, потягами і т.д; у сфері оборони, а саме у керування безпілотниками та роботизації хірургічних систем, аналізі зображень.

За типом керування виділяють дві класифікації: детерміновані та адаптивні.

Детерміновані системи керування – працюють за чітко визначеними алгоритмами та правилами без адаптації до змін середовища. До прикладу можна привести системи поливу на великих сільськогосподарських територіях.

Адаптивні системи керування – змінюють свої параметри залежно від зовнішніх умов. Інтелектуальні системи управління дорожнім рухом, які реагують на завантаженість трас можна віднести до цього типу.

Остання, проте не менш важлива класифікація – за архітектурою, а саме виділяють централізовані та децентралізовані системи.

Централізовані мають один головний контролер, що здійснює управління всією системою. Децентралізовані складаються з кількох незалежних модулів, які можуть працювати автономно або в координації між собою.

1.2 Методи та алгоритми керування рухомими об'єктами

Ефективне керування рухомими об’єктами базується на поєднанні математичного моделювання та застосування алгоритмів, які дозволяють забезпечити стабільність, точність та адаптивність функціонування системи. Існує кілька підходів до організації процесу керування, які класифікують за рівнем складності, наявністю адаптації до середовища та використанням математичних моделей.

Одним із найбільш поширених методів є PID-регулятор, який широко використовується для підтримки параметрів системи в заданих межах. Його принцип роботи базується на обчисленні похибки між бажаним і фактичним значенням, на яку реагують три складові регулятора: пропорційна забезпечує реакцію до похибки, інтегральна накопичує помилки за часом та компенсує зсув та диференціальна прогнозує зміну похибки.

Формула PID-регулятора має наступний вигляд:

$$u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de(t)}{dt}, \qquad (1.1)$$

де: $e(t)$ – різниця між бажаним і фактичним значенням параметра (похибка),

K_p – коефіцієнт пропорційного регулювання,

K_i – коефіцієнт інтегрального регулювання,

K_d – коефіцієнт диференціального регулювання [9].

Перевагами є простота реалізації, універсальність, висока точність за умов правильної настройки. До недоліків належить потреба в точному налаштуванні параметрів та слабка придатність до нелінійних систем або змін середовища.

Однак, сучасні тенденції застосування PID вимагають гнучкішого підходу. У складних системах PID часто виступає фінальним етапом у багаторівневих архітектурах — наприклад, при стабілізації об'єкта навколо траєкторії, яку

побудовано за допомогою методів планування, таких як A* чи MPC. У таких гібридних структурах класичні регулятори виконують роль локальних стабілізаторів, що компенсують шум, завади або похибки позиціонування.

Покращити точність керування у нелінійних системах можна, використовуючи альтернативу класичним методам – нечітку логіку, яка дозволяє працювати із складними та нелінійними системами, де матмодель є недостатньо точною.

Принцип роботи нечіткої логіки полягає у застосуванні лінгвістичних змінних та наборі правил, які послідовно перетворюються у чіткі значення через процес дефазифікації (рис.1.3). Розширяючи подану інформацію, важливо додати, що чітка логіка або стандартна зазвичай працює із полярними поняттями «1» та «0», проте нечіткі системи здатні прийняти середні значення «0,71» або «0.15», що можуть виражати таку стилістику, як «тепло», «помірно», «швидко».

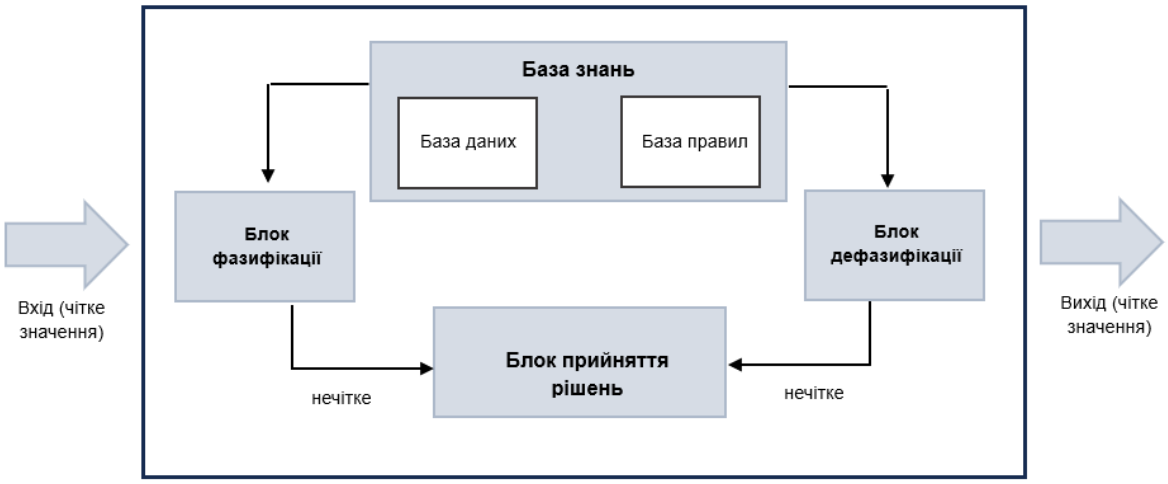

Рисунок 1.3 – Система нечіткого виведення

Основною перевагою нечіткої логіки виступає ефективна працездатність у складних та невизначених середовищах, проте найбільшим недоліком є важке налаштування правил, які зазвичай мають бути налаштовані вручну, а також велика кількість обчислювальних ресурсів, коли кількість правил може сягати тисячам.

Комбіновані PID-Fuzzy регулятори є прикладом гібридних систем, де традиційний підхід посилюється здатністю нечіткої логіки до адаптації.

З розвитком штучного інтелекту з’явилися адаптивні алгоритми, що дозволяють системам навчатися та самостійно налаштовувати параметри без необхідності ручного втручання. Одним із перспективних напрямів є використання методів машинного навчання, таких як Reinforcement Learning (RL) (рис. 1.4) [10], коли система навчається шляхом спроб і помилок, отримуючи винагороду за правильні дії .

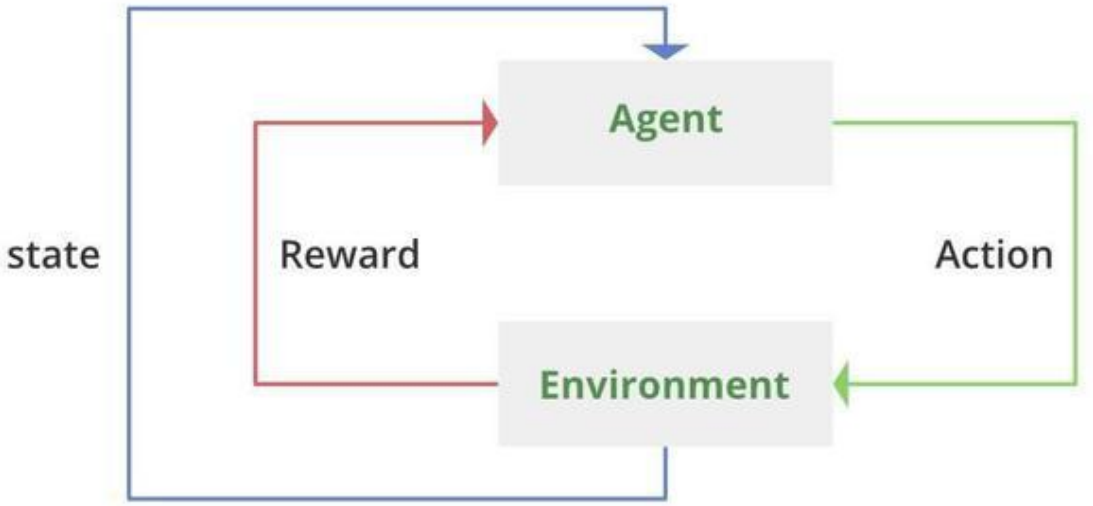

Рисунок 1.4 – Схема процесу навчання агента через взаємодію із середовищем, винагороду та зміну стратегії [10]

Переваги адаптивних систем:

- ✓ Пристосування до нових умов без попереднього програмування.
- ✓ Покращення продуктивності у складних середовищах.

Недоліки:

- – Велика кількість данних для навчання.
- – Нестабільність у критичних ситуаціях.

Формалізація завдання автономного керування зазвичай базується на кінематичних чи динамічних моделях. Кінематична модель описує рух без урахування сил, які його викликають, і часто використовують у завданнях планування траєкторій або орієнтації, тоді як динамічна модель включає масу, інерцію, сили та моменти, що дозволяє моделювати реальні умови руху.

Типовим підходом є представлення системи у формі нелінійної моделі стану:

$$\hat{x}(t) = f(x(t), u(t), t) \quad (1.2)$$

$$y(t) = g(x(t), u(t), t) \quad (1.3)$$

де $x(t)$ - вектор стану,

$u(t)$ - вектор керувальних дій,

$y(t)$ - вихід системи,

f, g - нелінійні функції, що описують динаміку систему.

Завдання автономного керування зводиться до визначення такого керуючого впливу $u(t)$, який забезпечить досягнення бажаного стану x при мінімізації певного критерію якості (наприклад, енергоспоживання, година руху, відстань до перешкод). У цьому контексті значну роль відіграють алгоритми планування траєкторії — процедури, які дозволяють обрати оптимальний маршрут у просторі з урахуванням обмежень.

Серед класичних алгоритмів планування виділяють алгоритм A*, Дейкстрі та RRT.

Алгоритм Дейкстрі гарантує знаходження найкоротшого шляху у графі без негативних ваг. Застосовується в дискретизованому просторі, наприклад, при навігації мобільних роботів у сітковій середовищі. Гарантує знаходження глобального мінімуму, але є повільним за великих розмірів простору.

Алгоритм A* — удосконалення методу Дейкстрі, що включає евристичну оцінку відстані до мети.

RRT (Rapidly-exploring Random Tree) — стохастичний алгоритм, який швидко досліджує простір конфігурацій, що робить його ефективним у високовимірних та складно структурованих средах. RRT* гарантує асимптотичну оптимальність за умови необмеженої години виконання.

Після побудови траєкторії наступним етапом є її оптимізація з урахуванням фізичних можливостей системи. Посеред усіх методів можна виокремити наступні (табл. 1.1):

MPC — метод передбачувального керування, що будує оптимальну послідовність керувальних дій, виходячи з поточного стану та математичної моделі об'єкта.

DDPG – представник методів глибокого навчання з підкріпленням, який дозволяє навчитися керувати системою у складних та невизначених умовах без явно заданої моделі. Агент поступово вдосконалює свою політику дій, максимізуючи сумарне винагороду. Перевагою є здатність до навчання в умовах складної динаміки, однак такі моделі мають довгий цикл і можуть бути нестабільними без налаштування.

Таблиця 1.1

Порівняльна таблиця методів оптимізації

Метод керування	**Потреба в моделі**	**Адаптивність**	**Придатність до реального часу**	**Витрати обчислень**
PID	Необов'язкова	Низька	Висока	Низькі
MPC	Так	Середня	Обмежена	Високі
DDPG	Ні	Висока	Середня	Дуже високі

Вибір алгоритмів моделювання, планування та оптимізації залежить від складності середовища, обчислювальних можливостей та вимог до точності та надійності. У практичних реалізаціях часто використовують гібридний підхід — наприклад, спочатку виконується побудова маршруту методом A*, далі його згладжування MPC, а відстеження траєкторії — PID-регулятором з адаптивною компонентою.

1.3 Сучасні тенденції та виклики у сфері автономного керування

Вихід все нових та ефективних технологій на основі нейромереж свідчить про активне впровадження штучних систем в різні сфери діяльності, і системи керування не стають осторонь. Однак, поряд із технологічним прогресом постають нові виклики, зокрема питання етичності, безпеки та технічні бар'єри, які необхідно враховувати при впровадженні таких технологій, що буде також розглянуто в цьому блоці.

Штучні нейронні мережі (ШНМ) стали важливим інструментом у розробці автономних систем керування. Вони дозволяють системам не просто реагувати на зміну зовнішніх умов, а й прогнозувати майбутні ситуації на основі

накопичених даних. Окрім розглянутого раніше LR, використовуються також гібридні системи та методи глибокого навчання.

Глибоке навчання – застосовується для аналізу зображень та відео, що дозволяє, наприклад, дронам та автономним автомобілям ефективно розпізнавати об’єкти та уникати перешкод.

Гібридні системи, що є поєднанням класичних підходів та ШНМ, забезпечують баланс між традиційною керованістю та можливостями навчання, які, в свою чергу, покращують точність та швидкість реагування систем.

Незважаючи на високий потенціал, нейромережі мають обмеження, зокрема потребу у великих обсягах даних для навчання та труднощі з інтерпретованістю рішень.

Зі зростанням рівня автономності керованих систем постає питання відповідальності за їхні дії. Автономні автомобілі, військові дрони та медичні роботи можуть ухвалювати критично важливі рішення, що безпосередньо впливають на життя та безпеку людей. Проблеми відповідальності, алгоритмічні упередження та моральний вибір у критичній ситуації – етична проблематика сучасності.

Юридична відповідальність за дії автономних систем є складним і багатоаспектним питанням, яке наразі перебуває на стадії активного обговорення та формування як в Україні, так і в інших країнах. Основна проблема полягає у визначенні суб'єкта відповідальності за шкоду, заподіяну автономними системами, зокрема транспортними засобами.

В Україні питання відповідальності за шкоду регулюється, зокрема, статтею 1166 Цивільного кодексу України, яка передбачає, що майнова шкода, завдана неправомірними діями, підлягає відшкодуванню особою, яка її завдала [11]. Однак, у контексті автономних систем виникає питання: хто саме вважається відповідальною особою — виробник, власник чи оператор?

У статті "Відповідальність за шкоду, завдану автономними транспортними засобами" [12] зазначається, що рівень автоматизації транспортного засобу є ключовим фактором у визначенні відповідальності. Зокрема, для транспортних засобів з рівнями автоматизації від 0 до 2 відповідальність покладається на водія,

тоді як для рівнів 3 і вище питання відповідальності стає більш складним і може включати виробника або постачальника програмного забезпечення.

У європейських країнах також тривають дискусії щодо правового регулювання автономних систем. Зокрема, у сфері військових технологій, таких як автономні дрони, постають важливі правові та етичні питання щодо відповідальності за ненавмисну шкоду.

Окрім, безпеки та питання відповідальності, що є найбільш вагомим фактором використання АСК, не менш важливо розглянути технологічні бар'єри розвитку. Технічні обмеження найбільш активно гальмують повномасштабне впровадження таких систем у повсякденне життя.

Багато автономних систем використовують камери, лідари та радари, проте їх ефективність може знижуватися за складних погодних умов або у разі фізичних перешкод. Глибокі нейромережі та інші сучасні алгоритми вимагають значних обчислювальних ресурсів, що може бути критично важливим для роботи в реальному часі. А також постає питання інфраструктурного обмеження. Для ефективного функціонування автономного транспорту необхідно розвивати інтелектуальні дорожні системи, покращувати комунікацію між транспортними засобами та створювати захищені канали передачі даних.

Усі розглянуті в цьому підрозділі аспекти можуть бути піддані подальшому коригуванню, удосконаленню або нормативно-правовому закріпленню, що значно розширює перспективи розвитку автономних систем. Наукове співтовариство та розробники активно працюють над створенням відповідної законодавчої бази, яка врегулює питання відповідальності, безпеки та етичних аспектів використання автономних технологій.

Паралельно тривають дослідження у сфері квантових обчислень, які мають потенціал значно пришвидшити обробку великих масивів даних і мінімізувати затримки при прийнятті рішень у реальному часі. Важливим напрямом є також вдосконалення сенсорних технологій, що дозволить автономним системам забезпечувати більш точне та ефективне сприйняття навколишнього середовища, підвищуючи їхню адаптивність до складних і змінних умов експлуатації.

Висновки за розділом 1

У першому розділі було проведено огляд та аналіз автоматизованих систем керування рухомими об'єктами. Розглянуто їх класифікацію за рівнем автономності, архітектурою, сферою застосування та типами керування. Детально описано методи керування, серед яких особливу увагу приділено PID-регуляторам, нечіткій логіці та алгоритмам машинного навчання. Окремо проаналізовано підходи до планування траєкторій, включаючи A*, RRT та методи оптимізації (MPC, DDPG). Також висвітлено сучасні тенденції, пов’язані з впровадженням штучного інтелекту в автономні системи та пов’язані з цим етичні й технічні виклики. Узагальнено, що ефективне автономне керування можливе за умови інтеграції класичних регуляторів з інтелектуальними алгоритмами в рамках гнучких архітектур керування.

РОЗДІЛ 2
ТЕХНІЧНІ ОСНОВИ АВТОМАТИЗОВАНИХ СИСТЕМ КЕРУВАННЯ РУХОМИМИ ОБ'ЄКТАМИ

Для глибшого розуміння сутності автоматизованих систем керування необхідно звернутися до їхньої архітектури — структурної основи, що об'єднує апаратні та програмні компоненти в єдине функціональне ціле. Саме через призму архітектури можна оцінити здатність системи до обробки інформації, ухвалення рішень і автономної дії в реальному часі.

Архітектура автоматизованих систем зазнала суттєвих змін протягом десятиліть. Від простих механізованих пристроїв епохи промислової революції до складних, інтегрованих систем сучасності — масштаб і вплив автоматизації значно зросли. У 1940–1950-х роках основу автоматизації складали релейні системи, які виконували прості, повторювані операції в межах виробництва (табл. 2.1).

З появою комп’ютерної техніки та цифрових технологій у 1970-х роках відкрився новий етап розвитку – системи керування почали впроваджувати програмовані логічні контролери (PLC) та розподілені системи керування (DCS), що означало перехід до більш гнучкої промислової автоматизації.

У 1990-х роках значну трансформацію архітектури забезпечив розвиток інтернету, який відкрив можливості для віддаленого моніторингу й керування. Період ознаменувався впровадженням систем SCADA (наглядове керування і збір даних), що стали основою інтелектуальніших та більш взаємопов’язаних рішень.

Сьогоднішні архітектури автоматизованих систем схиляються до кіберфізичних підходів, використання технологій Інтернету речей (IoT) і штучного інтелекту (ШІ), що забезпечує високий рівень автономності, адаптивності та інтеграції в реальному часі [13].

Таблиця 2.1

Розвиток автоматизації

Ера	Розвиток автоматизації
1940-1950-ті роки	Прості релейні системи для повторюваних завдань.
1970-ті роки	Впровадження DCS і PLC.
1990-ті роки	Інтернет-технології, SCADA системи.
сучасність	Кіберфізичні системи, IoT, рішення на основі ШІ.

Такий еволюційний розвиток автоматизованих систем керування не лише свідчить про ускладнення їхньої структури, але й актуалізує потребу в розумінні ключових складових, які формують архітектуру сучасних АСК. У центрі цієї архітектури — тісна взаємодія апаратних і програмних компонентів, які спільно забезпечують збір даних, ухвалення рішень та фізичну реалізацію керувальних дій.

2.1 Апаратна архітектура автоматизованих систем керування

Апаратна частина автоматизованої системи є фундаментом, на якому ґрунтується вся логіка керування. Вона включає низку фізичних елементів, які забезпечують безперервну взаємодію із середовищем, збір інформації, обробку сигналів та виконання керувальних дій. Основними елементами апаратної складової є сенсори, контролери, приводи, а також допоміжні засоби зв’язку та живлення.

Сенсори є «органами чуття» системи. Вони здійснюють вимірювання фізичних параметрів об'єкта або навколишнього середовища та перетворюють їх у цифрові або аналогові сигнали для подальшої обробки. Залежно від призначення системи використовуються різні типи датчиків (рис. 2.1):

- Позиційні — визначають координати або положення об’єкта (енкодери, GPS-модулі).

- Швидкісні — вимірюють швидкість або кутову швидкість (тахометри, гіроскопи).
- Акселерометри — фіксують прискорення та коливання.
- Оптичні — розпізнають об'єкти або перешкоди (лідар, камери, інфрачервоні сенсори).
- Екологічні — фіксують температуру, вологість, тиск, гази тощо.
- Сенсори сили/моменту — використовуються, зокрема, в робототехніці для контролю взаємодії із зовнішніми об’єктами [14].

Рисунок 2.1 – Приклади сенсорів

Контролер — це електронний пристрій або модуль, який здійснює обробку вхідних сигналів від сенсорів або інших пристроїв, приймає рішення на основі заданого алгоритму та формує вихідні сигнали для керування виконавчими механізмами або іншими компонентами системи. У промисловій автоматизації зазвичай застосовуються три основні типи контролерів: програмовані логічні контролери, розподілені системи керування, згадані раніше, а також програмовані контролери автоматизації.

PLC спочатку були розроблені як альтернатива релейній логіці, зокрема в автомобільній промисловості. З часом вони зазнали модернізації, отримавши підтримку аналогових входів і виходів, що дозволило значно розширити сферу їх застосування за межі лише дискретної автоматизації. Він підходить для керування рухомими об'єктами, особливо в умовах дискретності, простої або середньої складності задач, або із жорсткими вимогами до надійності. Проте із задачами високої динаміки - гнучкої адаптації або навчання – можуть виникати труднощі.

Своєю чергою, DCS були розроблені для керування безперервними технологічними процесами, такими як переробка нафти чи очищення води. Хоча сучасні PLC також здатні виконувати багато з функцій DCS, останні зазвичай забезпечують тіснішу інтеграцію з інтерфейсами операторів та засобами інженерної конфігурації. Рідше використовується саме для керування рухомими об’єктами, оскільки більше підходить для безперервних технологічних процесів, де об'єкти є стаціонарними, а також не розраховані на високо динамічні або автономні мобільні системи.

Програмовані контролери автоматизації представляють собою наступний етап розвитку промислових контролерів, які поєднують функціональні можливості PLC та DCS, але при цьому базуються на більш вагомих обчислювальних платформах. На відміну від класичних контролерів із закритою архітектурою, PAC мають гнучку, модульну структуру та за своїми можливостями наближені до персональних комп’ютерів, хоча і пристосовані до експлуатації в умовах промислового середовища [15].

PAC є оптимальним рішенням для складних та високо динамічних систем, таких як:

- Автономні транспортні засоби (наприклад, безпілотники, мобільні платформи, автоматизовані візки)
- Системи з великою кількістю сенсорних входів, що потребують високої швидкості обробки та адаптивного реагування.

- Задачі, які передбачають інтеграцію з комп’ютерним зором, технологіями машинного навчання, Інтернетом речей (IoT) та інтелектуальними алгоритмами керування.

Приводи — це елементи системи (рис. 2.2), які забезпечують безпосередній фізичний вплив на об’єкт керування . Вони перетворюють керуючі сигнали від контролера в механічний рух або іншу дію. За типом енергії, що використовується, приводи поділяються на електричні, пневматичні, гідравлічні та комбіновані (рис. 2.3).

Рисунок 2.2 – Типи приводів

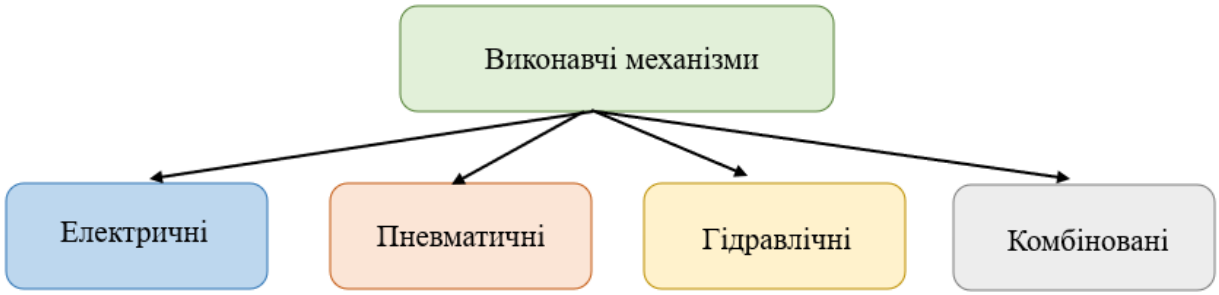

Рисунок 2.3 – Класифікація виконавчих механізмів

Електричні приводи найбільш використовувані завдяки своїй точності, простоті інтеграції та контролю. Поділяються вони на приводи прямого та крокового струму, сервоприводів та безколекторні двигуни.

DC-приводи забезпечують просте регулювання швидкості та напряму обертання, широко застосовуються в мобільній робототехніці. Stepper-мотори дозволяють точно позиціонувати об'єкт без зворотного зв’язку. Ідеальні для задач, де потрібно точне покрокове переміщення. Сервоприводи мають вбудовану систему зворотного зв’язку (зазвичай енкодер), що дозволяє досягати високої точності, стабільності та швидкодії. Безколекторні двигуни поєднують високу ефективність і довговічність, застосовуються в дронах, електротранспорті.

Пневматичні приводи працюють за рахунок стисненого повітря. Зазвичай використовуються для простих дій: натискання, піднімання, переміщення на обмежену відстань. Характеризуються швидкодією, але обмеженою точністю.

Гідравлічні приводи забезпечують велику силу при компактних розмірах. Використовуються в важких мобільних системах, наприклад у військовій або будівельній техніці. Недоліком є складність керування та необхідність обслуговування гідравлічної рідини.

Лінійні актуатори здійснюють лінійний рух, тобто переміщення вздовж прямої. Можуть бути електричними, гідравлічними або пневматичними. Застосовуються в автоматичних дверях, регулюванні положення, підйомних механізмах тощо .

У структурі апаратного забезпечення автоматизованих систем керування важливе місце займають допоміжні блоки, які забезпечують стабільну, безпечну та безперебійну роботу основних функціональних компонентів. Хоча ці елементи не здійснюють безпосереднього керування рухомими об’єктами, їхнє значення є критичним для забезпечення надійності всієї системи.

Перш за все, це стосується систем живлення, які відповідають за подачу стабілізованої електроенергії до всіх елементів системи. До них належать блоки живлення постійного або змінного струму, стабілізатори напруги, а також акумулятори та джерела безперебійного живлення, що забезпечують автономність у випадках відключення енергопостачання. В особливо чутливих або мобільних системах, наприклад безпілотних платформах, застосовуються компактні літій-іонні батареї з інтелектуальними системами керування зарядом.

Не менш важливою є наявність систем захисту, які запобігають виходу з ладу обладнання внаслідок електричних або механічних впливів. Серед них можна виділити запобіжники, автоматичні вимикачі, TVS-діоди, а також оптоізолятори, що забезпечують гальванічну розв'язку між чутливими електронними елементами.

Окрему роль відіграють блоки індикації та діагностики, які використовуються для моніторингу стану системи, виявлення несправностей та оперативного реагування. До них належать світлодіодні індикатори, дисплеї, а також вбудовані діагностичні модулі, які проводять самотестування при запуску системи та під час її роботи.

2.2 Програмна архітектура автоматизованих систем керування

Програмна частина формує інтелектуальну основу на базі апартної частини, забезпечуючи логіку керування, обробку вхідних даних, реалізацію алгоритмів ухвалення рішень та оптимізацію траєкторій. Основними компонентами програмної частини виступають операційні системи реального часу (ОСРЧ), алгоритми керування, модулі адаптацій та безпеки і т.д.

У процесі еволюції операційні системи реального часу (ОСРЧ) формувалися на основі кількох основних архітектурних підходів, кожен із яких має свої переваги та обмеження.

Однією з перших реалізацій стала монолітна архітектура, згідно з якою операційна система розглядається як цілісний набір взаємопов'язаних модулів, що функціонують у межах ядра. Ці модулі забезпечують прикладному програмному забезпеченню інтерфейси для взаємодії з апаратним забезпеченням. Основним недоліком цього підходу є складність у прогнозуванні поведінки системи, що зумовлено тісною взаємозалежністю між модулями та складністю їхньої взаємодії.

Іншим підходом є рівнева або шарова архітектура, яка передбачає можливість звернення прикладного програмного забезпечення до апаратури як через ядро операційної системи та її сервіси, так і безпосередньо. У порівнянні з монолітною структурою, така архітектура забезпечує вищу передбачуваність

реакцій системи та сприяє більш швидкому доступу до апаратних ресурсів. Проте серед недоліків слід відзначити обмежену підтримку багатозадачності.

Сучасні ОСРЧ часто використовують архітектуру типу «клієнт-сервер», яка базується на принципі винесення системних сервісів на рівень користувача у вигляді окремих серверних процесів. У такій моделі мікроядро виконує функції диспетчера повідомлень, забезпечуючи взаємодію між користувацькими клієнтами та системними серверами (рис. 2.4).

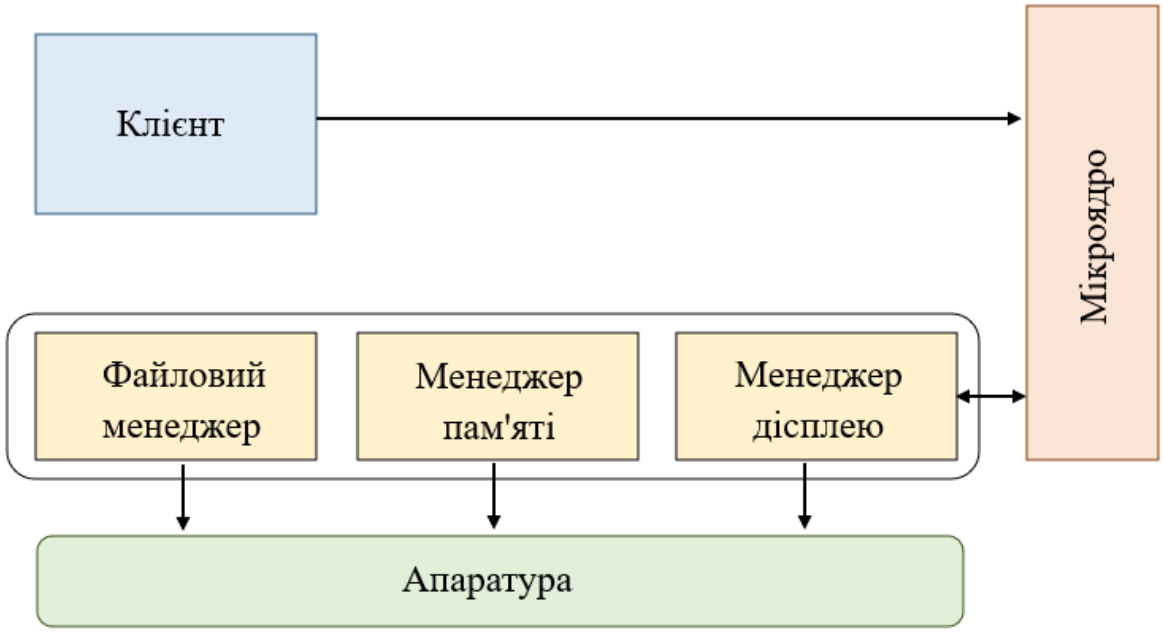

Рисунок 2.4 – Архітектура типу «клієнт-сервер»

До ключових переваг цього підходу можна віднести підвищену надійність, оскільки кожен сервіс працює автономно, що полегшує процес відлагодження та виявлення помилок; добру масштабованість, яка дозволяє відключати непотрібні сервіси без впливу на загальну працездатність системи; а також високу відмовостійкість — у разі зависання окремого сервісу його можна перезапустити без необхідності повного перезавантаження всієї операційної системи.

Операційні системи загального призначення (ОСЗП) здатні забезпечувати функціонування багатьох із зазначених вище сервісів. Водночас принципова відмінність сервісів, реалізованих у ядрах операційних систем реального часу, полягає в їхній жорсткій детермінованості, тобто у функціонуванні на основі

точного та гарантованого контролю часу. У даному випадку під детермінованістю мається на увазі те, що кожна операція або сервіс виконуються у межах фіксованого, наперед визначеного інтервалу часу, що забезпечує стабільну й передбачувану поведінку системи в умовах обмежених часових ресурсів (табл. 2.2).

Таблиця 2.2

Порівняльна таблиця ОСРЧ із ОСЗП.

Критерій	ОСРЧ	ОСЗП
Мета	Забезпечення своєчасного реагування на події, що виникають у процесі функціонування апаратного забезпечення.	Раціональний розподіл апаратних ресурсів комп'ютера між користувачами та прикладними задачами.
Фокус діяльності	Спрямовано на обробку зовнішніх подій, зокрема тих, що пов'язані з роботою обладнання.	Орієнтовано на взаємодію з користувачем та виконання його дій.
Призначення та позиціонування	Розглядається як інструмент для побудови спеціалізованих апаратно-програмних комплексів реального часу.	Сприймається як універсальна програмна платформа з готовими до використання прикладними засобами.
Цільова аудиторія	Розрахована на досвідчених розробників та інженерів.	Призначена для користувачів із середнім рівнем підготовки.

У межах реалізації програмної архітектури автоматизованих систем управління особливу роль відіграють програмні фреймворки та середовища розробки, що виконують функцію інтеграції логіки управління, взаємодії з апаратною платформою, симуляції поведінки об'єктів у віртуальному середовищі та організації зв'язку між окремими модулями. Найбільш поширеним і універсальним середовищем в галузі робототехніки та автономних систем є ROS (Robot Operating System) [16], що забезпечує модульну архітектуру, гнучке керування потоками даних та підтримку великої кількості апаратних конфігурацій. Завдяки концепції topics, services & actions, ROS дозволяє незалежно розробляти та тестувати функціональні блоки. Прикладом можуть виступати планувальник траєкторії, модуль виявлення перешкод або блок локалізації.

Паралельно з ROS активно застосовуються й інші інструменти, зокрема MATLAB/Simulink [17], що дозволяють моделювати поведінку керованих об'єктів на рівні диференціальних рівнянь, створювати цифрові контролери, реалізовувати симуляції та генерувати готовий код для вбудованих платформ. LabVIEW, у свою чергу, пропонує графічне середовище для швидкого прототипування систем з великою кількістю сенсорних входів та керувальних виходів, що зручно при розробці прототипів промислового призначення. Важливою тенденцією останніх років є перехід до використання інтерпретованих мов високого рівня, таких як Python, особливо у випадках, коли потрібна інтеграція з модулями машинного навчання або хмарними сервісами.

Завдання низькорівневого управління, де критично важливою є швидкодія і контроль над ресурсами, перевага все ще віддається мовам C та C++, які залишаються основними інструментами для програмування мікроконтролерів, драйверів сенсорів та вбудованих обчислювальних блоків.

Протоколи обміну даними відіграють ключову роль у процесах забезпечення взаємодії між програмними модулями, а також між програмним забезпеченням і апаратною платформою. Для внутрішньої комунікації між контролерами або між окремими модулями в межах однієї машини найчастіше застосовується протокол CAN (Controller Area Network), який забезпечує високу надійність, детермінізм передачі даних та підтримку пріоритетності. Тоді ж, коли система функціонує як частина промислової мережі або взаємодіє зі SCADA-системами, переважно використовують Modbus та OPC UA [18].

У контексті IoT-орієнтованих рішень, особливо для розподілених систем, все ширше використовують MQTT — легкий брокерний протокол, оптимізований для обмежених каналів зв'язку та енергоефективних пристроїв. Сучасні розробки на базі ROS 2 також передбачають використання DDS — високопродуктивного протоколу з підтримкою QoS, що дозволяє налаштовувати критичність, частоту оновлення та надійність передавання для кожного конкретного типу даних (рис. 2.5).

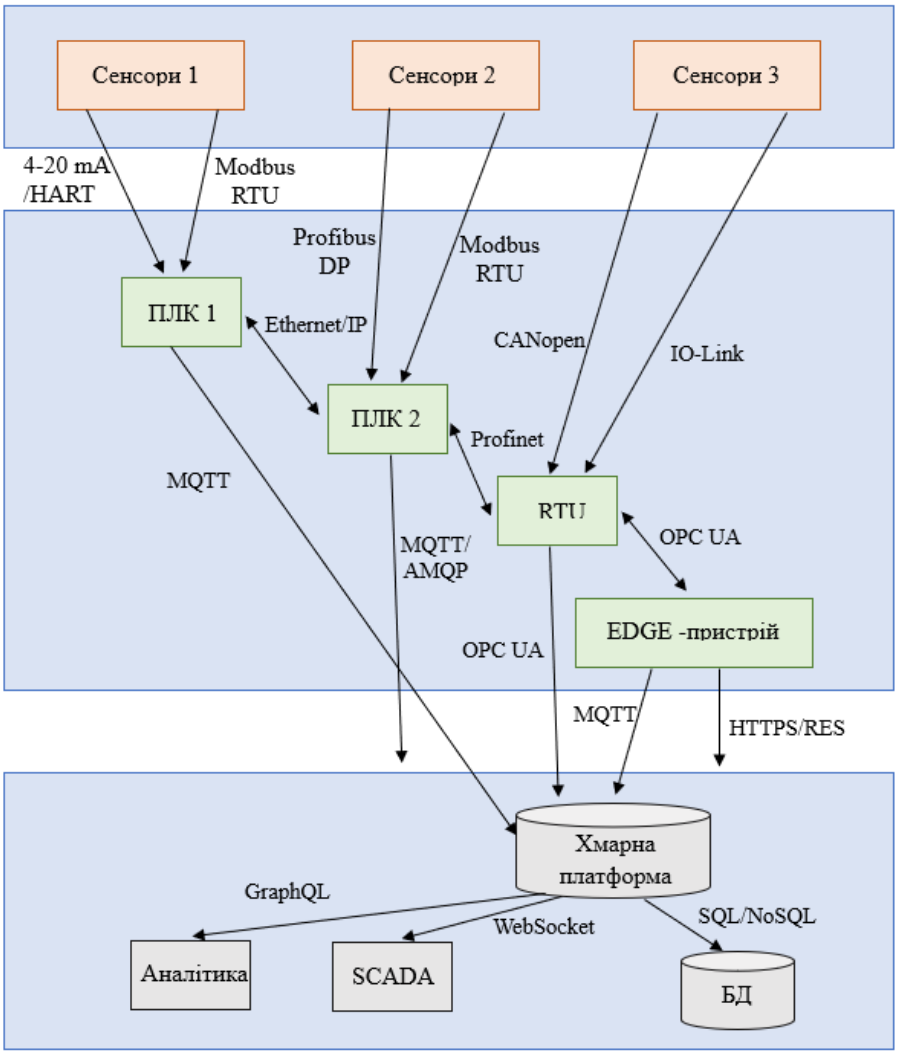

Рисунок 2.5 - Схема протоколів взаємодії в АСК між сенсорами, контролерами та хмарою

Розглядаючи архітектуру взаємодії між компонентами автоматизованої системи управління, доцільно представити узагальнену схему комунікаційних протоколів, які забезпечують передачу даних між сенсорними пристроями, ПЛК, периферійними обчислювальними модулями та хмарними платформами. Така структура дає змогу реалізувати гнучку, масштабовану та відмовностійку програмну архітектуру, яка інтегрує фізичний рівень об'єкта з цифровим рівнем управління та аналітики.

Важливою частиною життєвого циклу програмного забезпечення для АСК є етап віртуального тестування та відлагодження. З цією метою широко використовують симуляційні середовища, які дають змогу імітувати фізику

руху, вплив зовнішніх факторів, а також поведінку сенсорних систем без потреби використовувати реальні апаратні прототипи. Найбільш відомими середовищами є Gazebo – інтегрований з ROS; CARLA – для автомобільних симуляцій з високою фотореалістичністю; Webots – для швидкого прототипування роботів та AirSim (орієнтований на дрони та літаючі платформи) (рис. 2.6). Вони дозволяють розробникам тестувати алгоритми планування, локалізації, ухилення від перешкод чи навіть системи навчання з підкріпленням у безпечній та контрольованому середовищі.

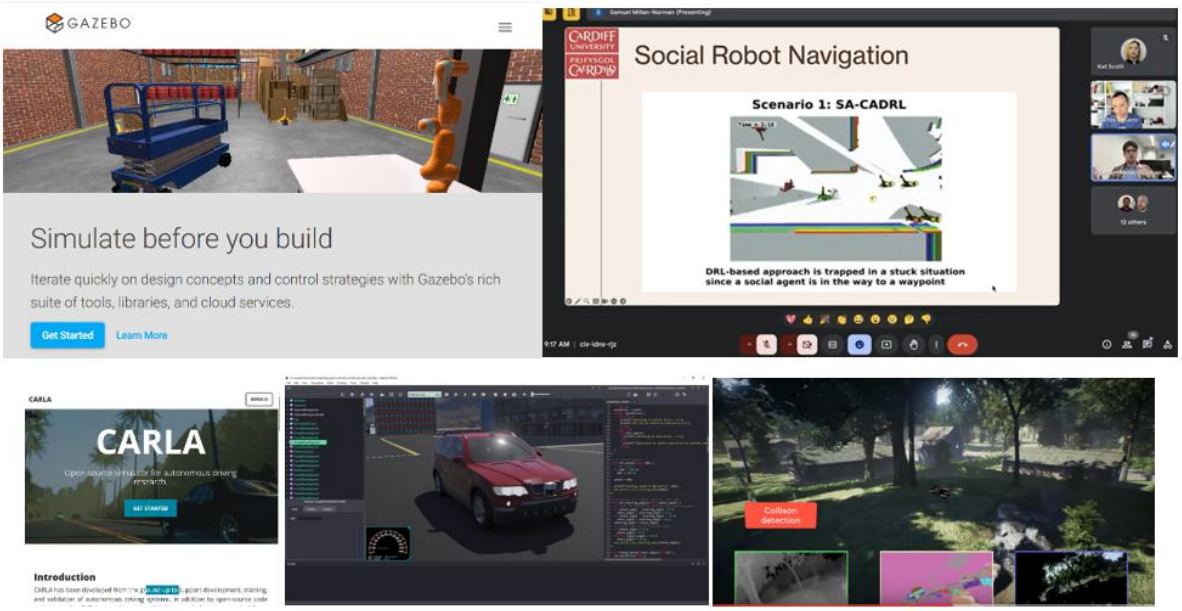

Рисунок 2.6 – Інтерфейси та проєкт реалізовані в середовищах

2.3 Синергія цифрових двійників та машинного навчання в системах управління

Один із перспективних напрямів застосування МН в автоматизованих системах керування — інтеграція з концепцією цифрових двійників (ЦД) – віртуального відображення фізичної системи, яке відтворює її стан, динаміку і поведінку в режимі реального часу, синхронізуючись із сенсорними даними.

ЦД може виступати як віртуальне середовище для безперервного навчання агентів машинного навчання, що дозволяє виконувати симуляції потенційних сценаріїв, оцінювати ризики і тестувати стратегії без ризику для реального об’єкта. У контексті RL, використання ЦД значно прискорює цикл навчання,

оскільки симуляція може бути проведена із більшою частотою, ніж фізичний процес.

Крім того, цифрові двійники сприяють підвищенню точності моделей МН, оскільки забезпечують генерацію великого обсягу синтетичних даних, які можуть бути використані для донавчання або перевірки нейронних мереж [19].

Ще одним практичним аспектом є використання для виявлення відхилень у поведінці фізичного об’єкта. Порівнюючи очікувану поведінку з фактичною, система може визначати аномалії, які вказують на несправність сенсорів, деградацію виконавчих механізмів або зміну навколишніх умов. МН в цьому випадку використовується для класифікації типів відхилень і прийняття оптимальних рішень щодо реагування.

Порівнюючи МН та ЦД (табл. 2.4) як інструменти, доцільно відзначити, що кожен із підходів має власну сферу оптимального застосування, проте обидва відіграють критично важливу роль у створенні інтелектуальних і адаптивних систем. Машинне навчання орієнтоване насамперед на виявлення закономірностей у даних, здатне адаптуватися до змін навколишнього середовища без попереднього програмування та приймати рішення на основі узагальненого досвіду. Цифрові двійники, у свою чергу, забезпечують високу точність моделювання фізичних об’єктів, їхню поведінку, деградацію або реакцію на зовнішні впливи. Вони дозволяють проводити верифікацію, діагностику та прогнозування з урахуванням конкретної архітектури об'єкта.

Основна відмінність полягає у фокусі: машинне навчання спрямоване на адаптацію до середовища через досвід, тоді як цифрові двійники — на відтворення поведінки системи через детальне моделювання.

Таблиця 2.4

Порівняльна таблиця МН та ЦД

Критерій	**Машинне навчання**	**Цифрові двійники**
Призначення	Самонавчання та адаптація на основі даних	Моделювання та симуляція фізичних систем
Джерело знань	Дані з сенсорів, логів, симуляцій	Фізичні моделі, технічна документація, дані
Залежність від реальних даних	Присутня	Помірна (може працювати зі штучними даними)
Придатність до прогнозування	Присутня (через моделі регресії/класифікації)	Присутня (через аналітичні/гібридні моделі)
Реагування на зміну стану	Адаптивне через навчання	Через модифікацію моделі або параметрів
Взаємодія з фізичним об’єктом	Опосередкована (через обчислення/рішення)	Тісна, із синхронізацією в реальному часі
Технічна складність реалізації	Висока (навчання моделей, налаштування)	Висока (створення точної цифрової копії)
Основна перевага	Гнучкість, здатність навчатися з досвіду	Реалістичне моделювання поведінки системи
Поведінка	Прийняття рішень на основі закономірностей	Верифікація, симуляція, прогнозування
Оптимальний сценарій	Динамічне середовище з невизначеністю	Критичні системи з високими вимогами до верифікації

У типовій архітектурі синергії ЦД синхронізується із сенсорними або телеметричними даними фізичної системи. Оновлена модель передає інформацію до МН-блоку, який, у свою чергу, може коригувати параметри керування, здійснювати прогноз майбутніх станів, або навіть переналаштовувати структуру контролера. У зворотному напрямку результати машинного аналізу впливають на віртуальні симуляції — наприклад, для тестування нових стратегій керування без ризику для реального об’єкта.

У другому розділі було розглянуто технічні основи побудови автоматизованих систем керування рухомими об’єктами, що охоплюють як апаратну, так і програмну складову. Систематизовано ключові елементи апаратної частини: сенсори, контролери, приводи та допоміжні компоненти, які забезпечують взаємодію системи із зовнішнім середовищем, збір і обробку

даних, а також виконання керувальних дій. Визначено роль сучасних типів контролерів та виконавчих механізмів у забезпеченні точності, адаптивності й надійності роботи в умовах динамічних змін.

Також було охарактеризовано програмну архітектуру як інтелектуальний рівень системи, що включає операційні системи реального часу, фреймворки, інструменти симуляції та протоколи обміну даними. Особливу увагу приділено забезпеченню детермінованості, масштабованості й адаптивності програмної частини, яка дозволяє реалізовувати автономну поведінку об’єкта та взаємодію з іншими системами.

Окремо розглянуто інтеграцію цифрових двійників і машинного навчання як перспективного напрямку розвитку автоматизованих систем. Визначено переваги їх поєднання для створення віртуального середовища навчання, прогнозування поведінки системи та адаптивного реагування на зміну зовнішніх умов. Така синергія підвищує точність, надійність і автономність сучасних систем керування та формує підґрунтя для створення самонавчальних і самокоригувальних рішень нового покоління.

Висновки за розділом 2

У другому розділі проведено ґрунтовний аналіз технічних основ автоматизованих систем керування рухомими об'єктами. Розглянуто еволюцію архітектур автоматизації – від релейних систем до сучасних кіберфізичних систем із використанням IoT та штучного інтелекту. Виділено ключові компоненти апаратної частини, зокрема сенсори, контролери (PLC, DCS, PAC), виконавчі механізми (електричні, пневматичні, гідравлічні приводи) та допоміжні блоки живлення й захисту. Окрему увагу приділено класифікації та застосуванню різних типів контролерів у динамічних системах.

Програмна архітектура розглянута як складова, що забезпечує реалізацію логіки керування, обробку даних і прийняття рішень. Показано роль операційних систем (реального часу й загального призначення), середовищ моделювання (Webots), протоколів зв’язку та підходів до розподіленої обробки даних. Акцент

зроблено на взаємодії апаратної та програмної частин у контексті забезпечення адаптивного, точного й ефективного керування в умовах змінного середовища.

Таким чином, у розділі сформовано цілісне уявлення про технічну базу, що є основою для побудови автономних систем керування та реалізації складних алгоритмів навігації та стабілізації.

РОЗДІЛ 3
РОЗРОБКА ТА МОДЕЛЮВАННЯ СИСТЕМИ КЕРУВАННЯ ДРОНОМ У СЕРЕДОВИЩІ WEBOTS

У межах роботи моделювання виконує функцію зв'язувальної ланки між теоретичними засадами систем автоматичного керування та практичним втіленням концепції автономного польоту. Його мета полягає в створенні цифрової експериментальної платформи для апробації підходів до побудови стабілізованої та керованої системи на прикладі безпілотного літального апарата.

Виходячи з аналізу сучасних тенденцій, викладених у вищезгаданих розділах, можна сформулювати низку прикладних задач моделювання:

- перевірка працездатності базового архітектурного підходу до організації системи керування дроном;
- дослідження взаємодії сенсорної підсистеми з блоком прийняття рішень;
- оцінка ефективності простих алгоритмів стабілізації та орієнтації в тривимірному середовищі;
- апробація механізмів підтримки заданої висоти, напрямку та автономного переміщення;

3.1 Віртуальний прототип дрона

Для реалізації зазначених задач обране середовище Webots [20], що забезпечує інтегровану підтримку фізичного моделювання, розширену архітектуру пристроїв та інтерфейси для написання користувацьких контролерів. У роботі використовується зразок квадрокоптера dji mavic2pro (рис. 3.1), що дозволяє зосередитись на логіці керування, не витрачаючи ресурси на побудову фізичної моделі з нуля. Поступове вдосконалення функціональності в межах шаблону ефективно ітеративно перевіряє вплив змін на стабільність та керованість системи.

Рисунок 3.1 – Віртуальний прототип дрона dji mavic2pro

Моделювання автономного квадрокоптера полягає у чіткому окресленні параметрів, що надходять до системи керування і результуючими у процесі роботи. До вхідних параметрів системи належать:

- орієнтація дрона в просторі;
- просторове положення;
- кутові швидкості обертання;
- телеметрична інформація про відхилення від цільового положення або траєкторії;
- сигнали від алгоритмів навігації або зовнішнього планувальника;

До вихідних параметрів належать:

- команди керування для кожного з чотирьох моторів;
- інформація про зміну стану системи;
- дані для журналювання;

Таким чином, система функціонує як контур зі зворотним зв'язком, у якому стан платформи визначається сенсорними модулями, а дія реалізується через керування моторами відповідно до алгоритмів, заданих у контролері.

Наступним логічним етапом є опис архітектури, в основу якої покладено наступні ключові підсистеми: сенсорна, виконавча та контролер.

Сенсорна підсистема включає інерційний модуль для отримання поточних кутів нахилу та орієнтації, GPS-модуль — для фіксації координат у глобальній системі, гіроскоп — для визначення кутових швидкостей, необхідних для стабілізації.

Виконавча система містить приводи – чотири незалежні мотори із можливістю встановлення швидкості обертання та зміни параметра thrust для кожного двигуна залежно від стану дрона.

Контролер реалізований на мові python як окремий модуль користувача в Webots та включає реалізацію PID-регуляторів по висоті, ролу та пітчу, а також містить логіку станів для реалізації фази зльоту, польоту до точки, розвороту, повернення та зависання.

3.2 Реалізація системи керування

Базове керування дроном виконується за допомогою простої схеми регулювання на основі пропорційного контролю (P-регулятора (Рисунок 3.2, 3.3)) для стабілізації параметрів орієнтації (Roll, Pitch, Yaw) та висоти (Z).

Рисунок 3.2 – Базовий P-регулятор

```
const double roll_input = k_roll_p * CLAMP(roll, -1.0, 1.0) + roll_velocity + roll_disturbance;
```

Рисунок 3.3– Вертикальний контроль

```
const double vertical_input = k_vertical_p * pow(clamped_difference_altitude, 3.0);
```

Керування здійснюється вручну з клавіатури, а в технічній складовій використовуються наступні пристрої:

- GPS для визначення координат (X, Y, Z),
- IMU (Inertial Unit) для отримання Roll, Pitch, Yaw,
- Gyroscope — для контролю кутової швидкості,
- LED — для візуального індикаційного зворотного зв’язку,
- Camera Motors — стабілізація оглядової системи (roll/pitch камери),
- Propeller Motors — чотири гвинти з індивідуальним керуванням швидкістю.

Управління здійснюється за формулою, представленою у Рисунок 3.4, де vertical_input формується з відхилення за висотою (використано кубічне підсилення помилки (Рисунок 3.3)), a roll_input, pitch_input і yaw_input залежать від відхилення кута та заданих збурень з клавіатури.

Рисунок 3.4 Алгоритм керування

```
motor_input = k_vertical_thrust + vertical_input ± roll_input ± pitch_input ± yaw_input
```

Для глибшого аналізу роботи системи стабілізації та польоту дрона було проведено логування параметрів. Дані зберігались у форматі CSV-файлу (табл.3.1) з такими ключовими параметрами: час, координати, кути орієнтації та цільова висота.

Таблиця 3.1

Логування інформації прольотів з базовими налаштуваннями

Time	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**	**TargetAltitude**	**ActualAltitude**
1.81	0.0053	0	0.0857	0	-0.0697	-3.1416	1.3	0.0857
2.61	0.0053	0	0.0857	0	-0.0697	-3.1416	1.3	0.0857
3.41	0.0102	-0.0002	0.0863	-0.0009	-0.0632	3.1381	1.3	0.0863
4.21	0.0534	0.0002	0.2792	0.0013	0.0093	3.0752	1.3	0.2792
5.01	0.0513	0.0089	0.7268	0.0013	0.0069	3.0389	1.3	0.7268
5.81	-0.1485	0.0426	1.2698	0.0013	0.0758	3.0183	1.3	1.2698
6.61	-0.6104	0.1076	1.6123	-0.0003	0.0714	3.0434	1.3	1.6123
7.41	-1.193	0.1552	1.721	0.0017	0.0995	-2.9938	1.3	1.721
8.21	-1.7911	0.1491	1.6906	0.0014	0.0531	-2.842	1.3	1.6906
9.01	-2.4489	0.0627	1.5751	0.001	0.0392	-2.8311	1.3	1.5751
9.81	-2.9704	-0.0242	1.4317	0.0013	0.0046	-2.8853	1.3	1.4317
10.61	-3.2934	-0.074	1.304	0.0013	0.0048	-2.9163	1.3	1.304

На основі отриманих даних, було побудовано графіки (рис.3.5 – 3.6). У стабільній системі значення Z має наближатися до цільового – 1.0 м і утримуватись із мінімальними коливаннями, проте в отриманих результатах дрон різко набирає висоту, що свідчить про перерегулювання. Після цього спостерігаються затухаючі коливання, але стабілізація на заданому рівні не досягається.

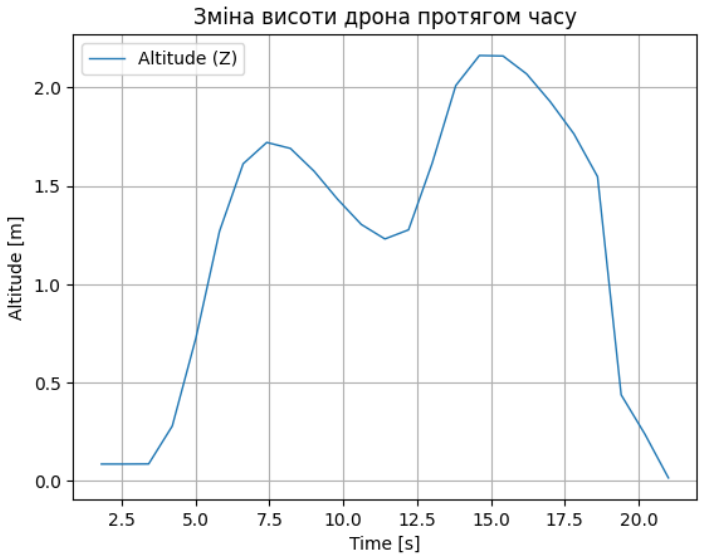

Рисунок 3.5 – Зміна висоти дрона протягом часу

Наступний графік (рис. 3.6) дає представлення про орієнтація дрона у просторі під час польоту. Значення Roll і Pitch змінюються при кожній спробі дрона стабілізуватись, причому з різкими стрибками. Yaw змінюється поступово, особливо при ручному введенні повороту. Спостерігається високий рівень шуму та нестійкість, особливо по осі Roll — це може вказувати на неузгоджену роботу PID-блоку або інерцію.

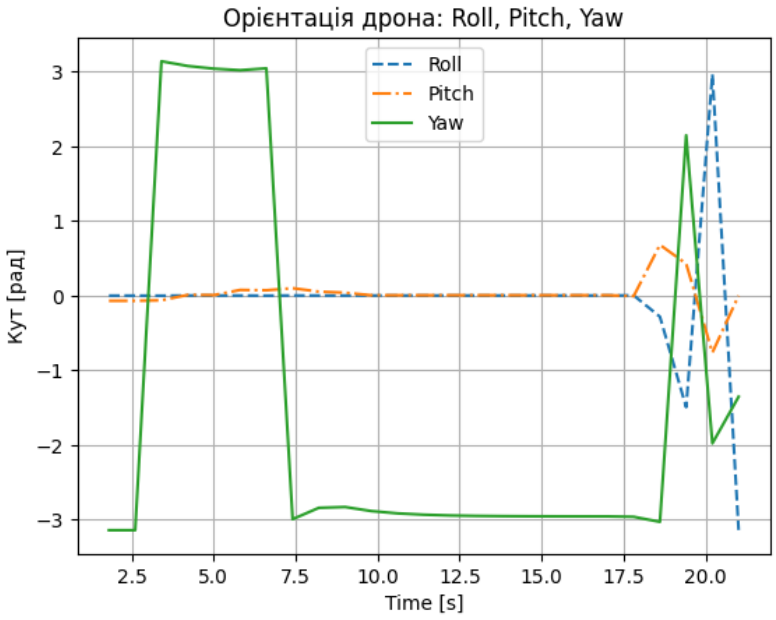

Рисунок 3.6 – Орієнтація дрона

У процесі тестування системи керування було виявлено, що під час різких змін напрямку руху дрон демонстрував нестабільну поведінку – у більшості випадків він не встигав відновити баланс, що призводило до втрати орієнтації та перекидання з переворотом вниз.

Наступним етапом у рамках роботи стало вдосконалення системи керування шляхом модифікації архітектури контролера та впровадження PID-регулятора (Рисунок 3.7). Рішення було обумовлене необхідністю забезпечення більш точного та стабільного керування дроном при маневруванні, особливо в умовах динамічних змін навколишнього середовища та команд користувача.

Параметр kp швидко реагує на похибку, ki усуває сталу похибку, kd зменшує осциляції та перерегулювання.

Рисунок 3.7 PID-регулятор

```
typedef struct {
  double previous_error;   // D-компонента (похідна)
  double integral;         // I-компонента: накопичена похибка
  double kp;               // коефіцієнт P
  double ki;               // коефіцієнт I
  double kd;               // коефіцієнт D
  double i_max;            // обмеження I-компоненти (анти-нагромадження)
} PIDController;
```

Ініціалізація контролера (Рисунок 3.8) відбувається для кожної осі з експерементально підібраними коєфіцієнтами.

Рисунок 3.8 Ініціалізація контролера

```
initPIDController(&roll_pid, k_roll_p, 2.5, 15.0, 10.0);
initPIDController(&pitch_pid, k_pitch_p, 1.5, 10.0, 10.0);
initPIDController(&vertical_pid, k_vertical_p, 0.15, 5.0, 5.0);
```

У результаті впровадження PID-регулятора в систему керування дроном спостерігається суттєве покращення стабільності та точності підтримки висоти, що підтверджується аналізом лог-файлів. Наведено діаграму (рис. 3.9), яка ілюструє зміну висоти дрона в часі при використанні PID-регулятора.

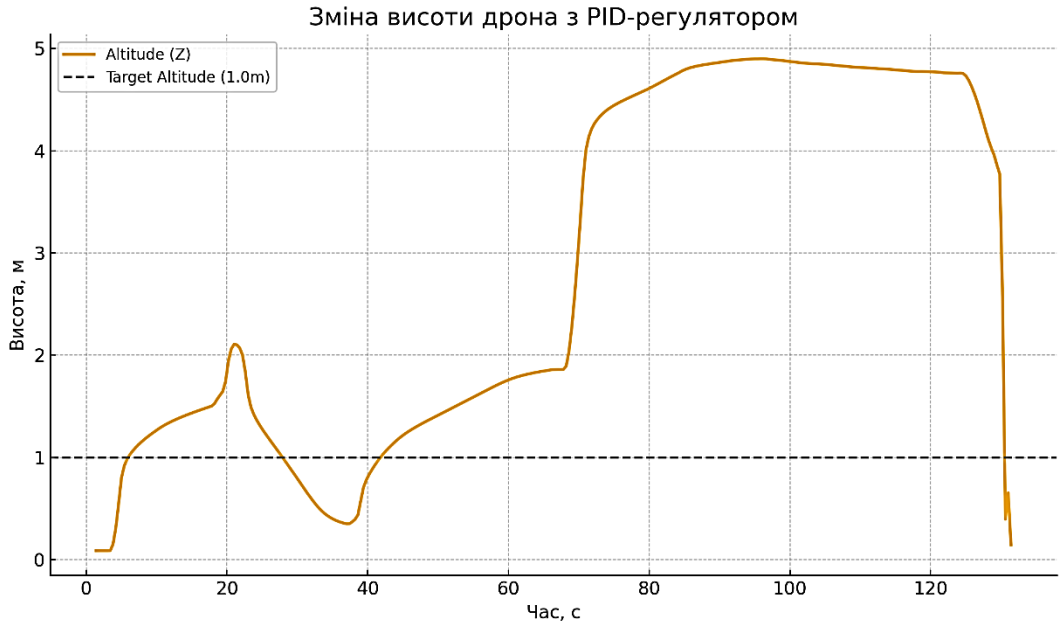

Рисунок 3.9 – Зміна висоти дрона

При використанні лише P-регулятора (рис. 3.10) висота дрона значно коливалась. Після зльоту відбувалося перерегулювання, і значення Z перевищувало цільову висоту (1.0 м), подекуди досягаючи 2.0 м і вище. Коливання були довготривалими, а стабілізація — повільною.

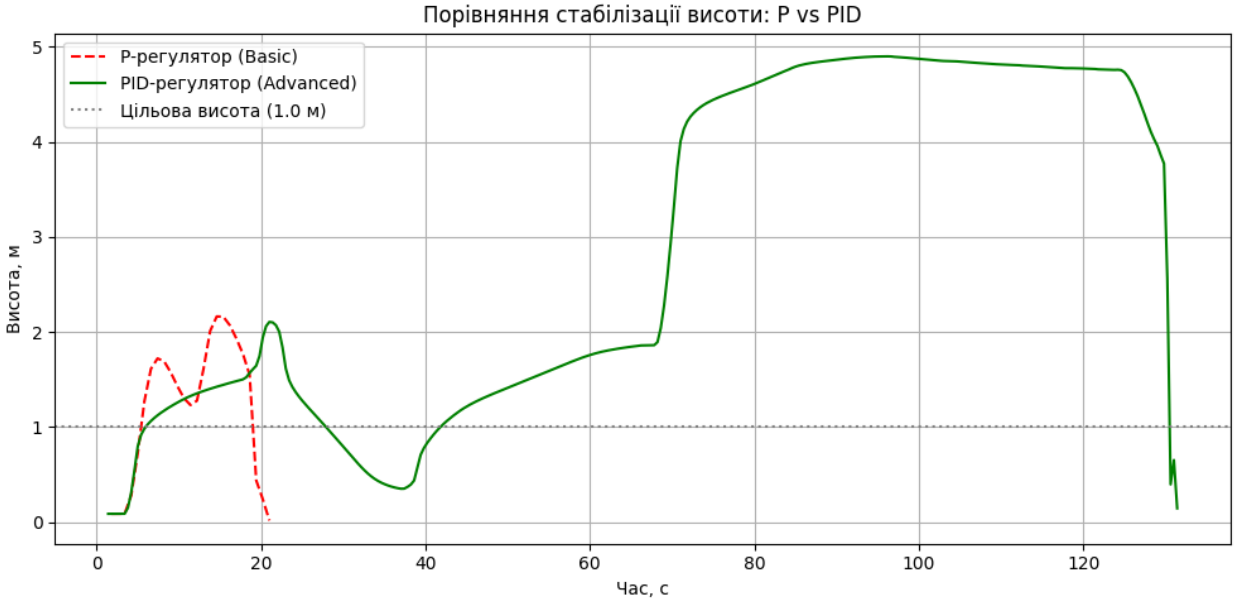

Рисунок 3.10 – Порівняння стабілізації висоти

На відміну від цього, при активації PID-регулятора поведінка дрона суттєво покращується. Завдяки інтегральній складовій контролер поступово компенсує залишкову похибку, а диференціальна — пригнічує коливання на

ранній фазі зльоту. У результаті дрон досягає цільової висоти значно швидше (менше ніж за 2 секунди) та стабільно її утримує з мінімальним відхиленням. Під час руху дрона вертикальні коливання згладжуються, і система демонструє надійність навіть при ручному керуванні.

3.3 Удосконалення системи керування

Після успішного тестування базових режимів стабілізації виникла потреба підвищити гнучкість системи керування. У попередніх версіях контролера дрон реагував виключно на заздалегідь задані команди без урахування умов навколишнього середовища.

Надалі буде представлено реалізацію автономного керування на основі автомата станів (FSM) із підтримкою виявлення та обходу перешкод, а також автоматичним поверненням у початкову точку. Така архітектура наближує розробку до реальних систем автономної навігації безпілотних літальних апаратів.

Нововведенням стала реалізація автомату станів, де кожен етап польоту подано як окремий логічний блок з чітко визначеними умовами переходу (табл. 3.2).

Таблиця 3.2

Логічні блоки

Стан	Дія	Умова переходу
TAKEOFF	Зліт до висоти target_altitude ≈ 15 м	досягнута висота → FLY_TO_A
FLY_TO_A	Політ до координат (-30, 20)	дрон в радіусі 0.5 м від точки A
FLY_TO_B	Політ до координат (-60, 10)	дрон в радіусі 0.5 м від точки B
RETURN_HOME	Автономне повернення до стартових координат	досягнута початкова позиція
LANDING	Плавне зниження до землі	висота < 0.2 м → завершення місії
AVOID_OBSTACLE	Обхід перешкоди (бічний маневр з утриманням висоти)	перешкода зникла → повернення до попереднього стану

Контролер поділений на окремі функціональні блоки:

- move_to_target() — обчислює напрямок і кут повороту, визначає, чи досягнуто цільову точку.
- detect_obstacle() — працює з front distance sensor або LIDAR, фіксуючи об’єкти на відстані менше 0.8 м.
- avoid_obstacle() — виконує просте ухилення праворуч через нахил з утриманням висоти.
- update_fsm() — обробляє зміни станів і реакції на середовище.

Перевагою такого підходу є масштабованість – функцію avoid_obstacle() можна замінити на складнішу поведінкову логіку або навіть використовувати нейронну мережу для планування дій.

Після реалізації базової архітектури контролера, основаної на автоматі станів, було проведено серію симуляційних тестів із записом даних про динаміку руху дрона. На основі отриманих логів побудовано графік (рис. 3.11) зміни орієнтації дрона в часі.

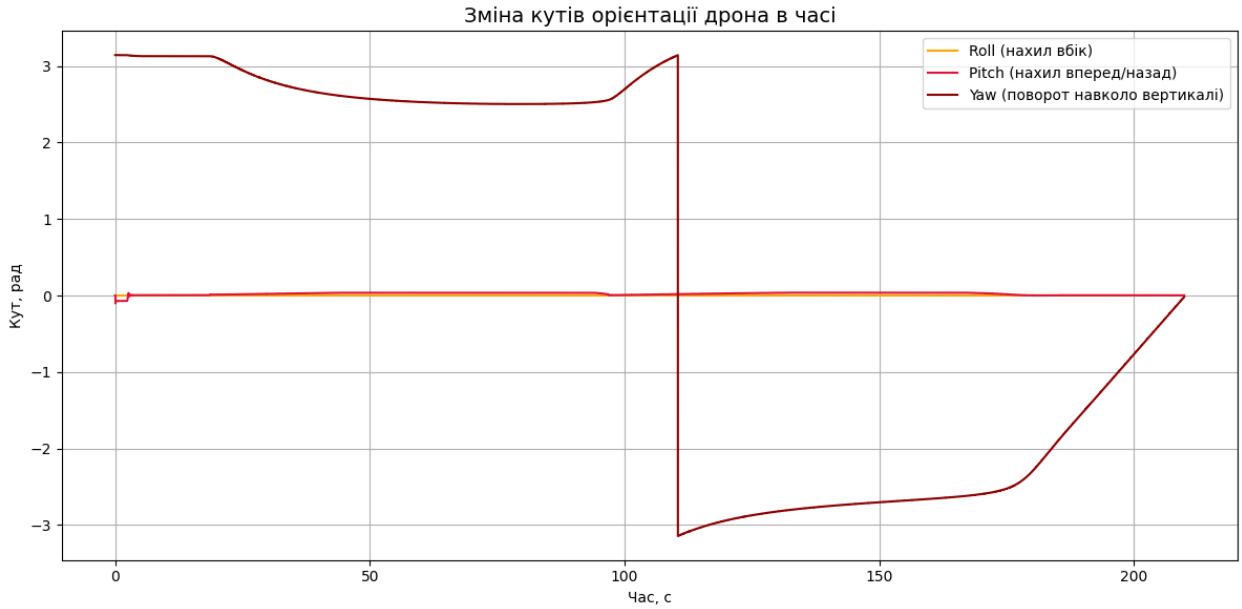

Рисунок 3.11 – Орієнтація руху дрона

Під час руху до заданих точок (FLY_TO_A, FLY_TO_B) спостерігаються помітні коливання значень yaw (курсу), що свідчить про активну корекцію напрямку польоту згідно з положенням цільової точки. Також зафіксовано

стабільність значень pitch — тангаж змінюється плавно, що є позитивною ознакою стабілізації по вертикальній осі. У режимі обходу перешкоди , хоча в поточній реалізації ухилення виконується лише бічним маневром, спостерігається характерний сплеск значення roll, який підтверджує активацію алгоритму ухилення праворуч. Під час автоматичного повернення та посадки кут yaw стабілізується, оскільки дрон прямує до початкової точки по прямій траєкторії, а значення висоти зменшуються плавно (табл. 3.3)

Таблиця 3.3

Лог-файли автономного керування

time	state	x	y	z	roll	pitch	yaw	fl	fr	rl	rr
363.82	RETURN_HOME	-60.23	9.61	15	0.001	0.002	-0.021	69.38287	69.52499	69.2442	69.34989
363.82	LANDING	-60.23	9.61	15	0.001	0.002	-0.02	69.38287	69.52499	69.2442	69.34989
363.83	LANDING	-60.23	9.6	14	0.001	0.002	-0.01	69.38165	69.38165	69.38165	69.38165

Загалом, поведінка дрона в симуляції відповідає очікуваній логіці автомату станів. Перехід між станами відбувається своєчасно, а стабілізація орієнтації дозволяє підтримувати контроль над апаратом під час усього польоту.

3.4 Автономне керування дроном на основі алгоритму A*

Після реалізації стабілізації та ручного керування у попередньому підрозділі, основним викликом стало забезпечення повної автономності польоту. Завдання полягало в тому, щоб дрон здійснив переліт між заданими точками — A (–30; 20), B (–60; 10), подальше патрулювання чотирьох контрольних точок, та безпечне повернення на стартову позицію, без участі оператора. У процесі навігації необхідно було враховувати як статичні (стіни, штучні об’єкти), так і динамічні перешкоди, дотримуючись вимог до безпечної висоти польоту (15 м), точності досягнення цілей (похибка < 1.5 м) та гарантованого повернення до точки зльоту.

Побудовано карту (рис. 3.12) розміром 100×100 клітинок (1 м/клітинка), яка покриває площу 100×100 м. Середовище створювалося з використанням

методів add_wall та add_obstacle, що дозволило моделювати як лінійні бар'єри, так і кругові зони перешкод.

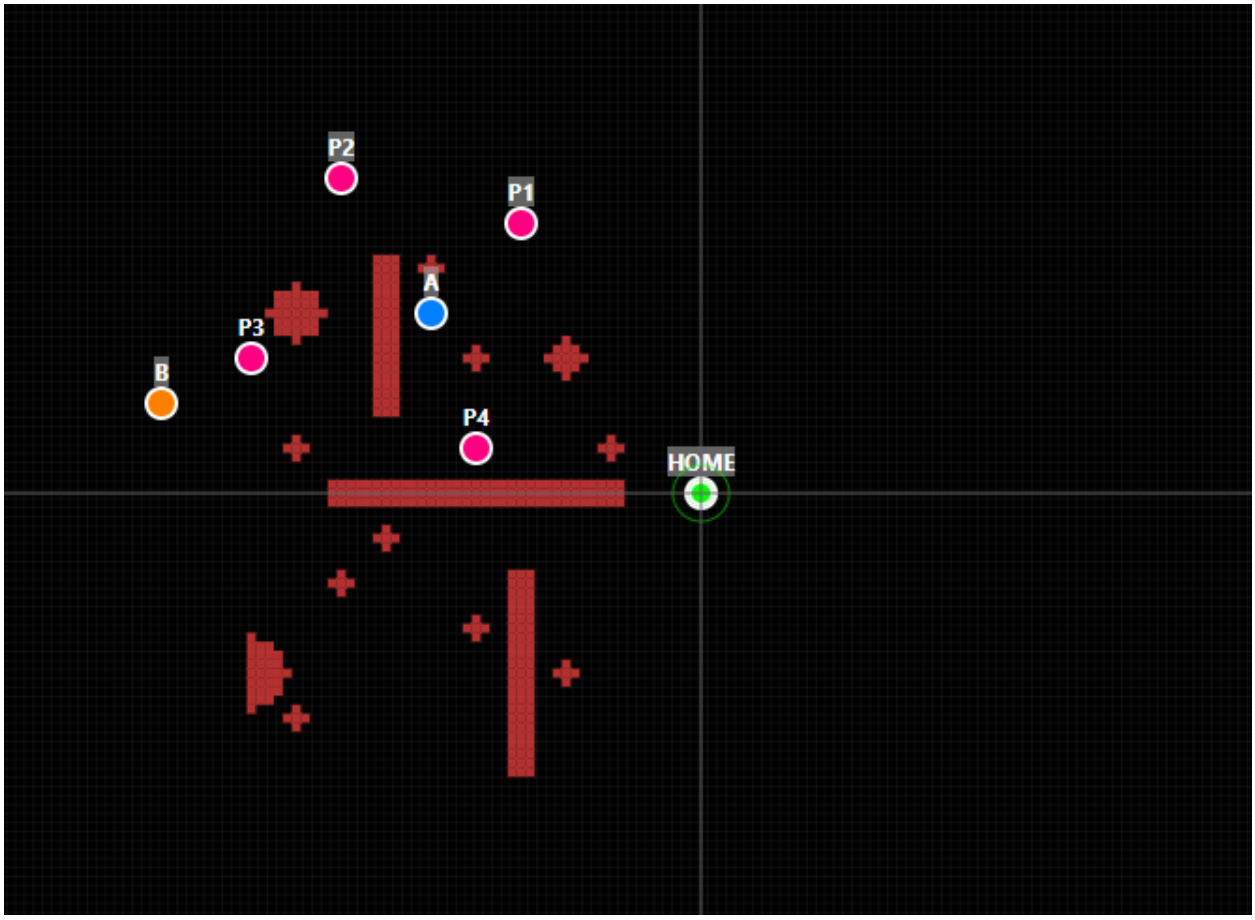

Рисунок 3.12 – Приклад побудованого середовища

Для планування маршруту обрано класичний A*-алгоритм з мангеттенською евристикою heuristic(a, b) (Рисунок 3.13). Старт і ціль конвертуються до координат сітки функцією world_to_grid. Алгоритм дозволяє 8-напрямкове переміщення з підвищеною вартістю діагональних кроків (1.41 проти 1.0). Завдяки використанню структури heapq, загальна складність оцінюється як O(N log N), що є прийнятною для карти розміром 10^4 клітинок.

У порівнянні з методом Дейкстри, A* суттєво зменшує кількість оброблюваних вузлів завдяки евристичній оцінці напрямку до мети. RRT був відкинутий через необхідність суворої відповідності цілі та неможливість отримання строго оптимального маршруту.

Планувальник A* було інтегровано у FSM-контролер, де кожен перехід стану (TAKEOFF → FLY_TO_A → … → RETURN_HOME → LANDING)

(рис.3.8) ініціює виклик функції recompute_path(). Обраний шлях заноситься у path_queue, що використовується для поетапного слідування. При виявленні перешкоди через detect_obstacle(), дрон переходить у режим AVOID_OBSTACLE.

Рисунок 3.13 A*-планувальник

```
def plan_path(self, start_xy, goal_xy):
    start = self.world_to_grid(start_xy)
    goal  = self.world_to_grid(goal_xy)

    open_set   = [(0, start)]
    g_score    = {start: 0}
    came_from  = {}

    while open_set:
        _, current = heapq.heappop(open_set)
        if current == goal:
            return self.reconstruct_path(came_from, current)

        for nx, ny in self.neighbours(current):
            tentative_g = g_score[current] + self.cost(current, (nx, ny))
            if tentative_g < g_score.get((nx, ny), 1e9):
                came_from[(nx, ny)] = current
                g_score[(nx, ny)]   = tentative_g
                f = tentative_g + self.heuristic((nx, ny), goal)
                heapq.heappush(open_set, (f, (nx, ny)))
```

Після звільнення шляху він повертається до попереднього стану. Крім того, для підвищення адаптивності середовища, реалізовано періодичне перепланування кожні 50 кроків (replan_interval = 50), а також тест на застрягання за допомогою is_stuck() — якщо зміщення занадто мале, виконується повторне планування.

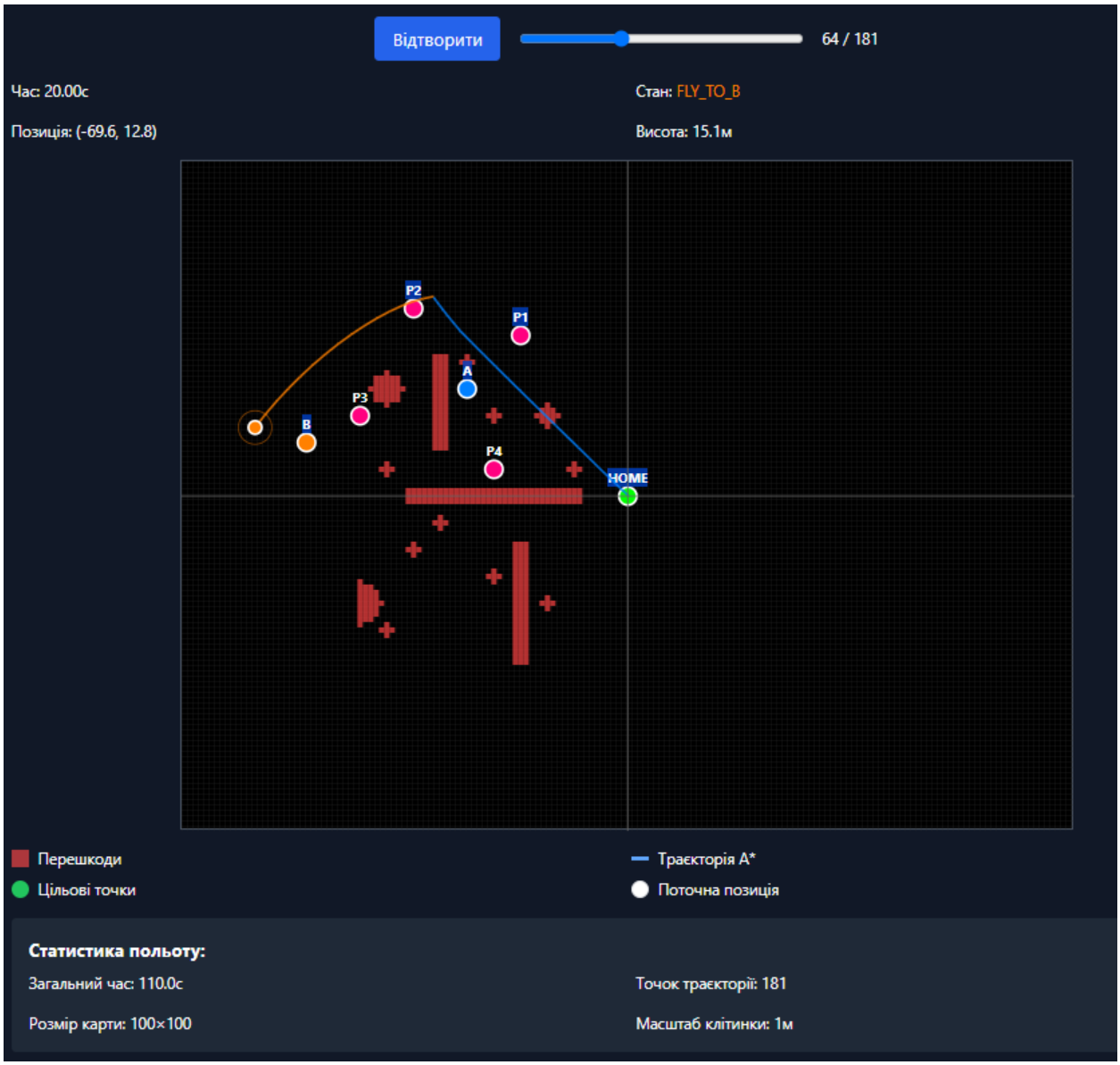

Рисунок 3.14 – Режим FLY_TO_B

Рух до цільового вузла реалізовано через обчислення yaw- та pitch-команд (compute_disturbances). Оцінюється кут між поточним напрямком і вектором до точки маршруту. При відхиленні менш ніж 45°, активується пропорційне прискорення вперед. Якщо відстань до цілі < 1.5 м — вузол вважається досягнутим і видаляється з черги. Усі рухи виконуються на фоновому стабілізаційному PID-контурі, наслідуваному з підрозділу 3.3.

Експеримент охоплював наступний сценарій: зліт → переліт до точок A та B → патрулювання (P1–P4) → повернення → посадка. Час переходу між ключовими станами: TAKEOFF→FLY_TO_A (3.02 с), FLY_TO_B→PATROL (27.02 с), повне завершення місії — 110.0 с.

A*-планувальник забезпечив точність навігації із середнім відхиленням 0.9 м при мінімальній відстані до перешкод 1.6 м, що гарантує безпеку (табл 3.4) . За весь маршрут (181 вузол) виконано лише 7 перепланувань, а середній час обчислення шляху склав 1.2 мс. Порівняно з ручним P-керуванням, час місії зменшився на ≈18 %, що демонструє ефективність A*-планувальника у завданнях навігації дрона.

Таблиця 3.4

Підсумкова статистика випробування

Показник	**Значення**
Загальний час місії	110,0 с
Кількість вузлів у глобальному шляху	181
Середня довжина переходу між вузлами	1,00 м
Розмір карти	100 × 100 м (1 м/клітинка)
Кількість перепланувань	7
Середній час A*-планування	1,2 мс
Середнє відхилення при досягненні цілі	0,9 м
Мінімальна відстань до перешкод	1,6 м

Алгоритм A* довів свою ефективність у задачах планування маршруту у сітковому середовищі, забезпечивши гарантовану побудову шляху та можливість адаптації у реальному часі. Обмеженням лишається 2D-природа сітки та фіксований масштаб.

Висновки за розділом 3

У третьому розділі було реалізовано повноцінну систему автономного керування дроном, яка охоплює всі ключові компоненти: стабілізацію, навігацію, планування маршруту та адаптацію до змін у середовищі. Розроблено FSM-контролер з підтримкою автономних переходів між фазами польоту, інтегровано алгоритм A* для побудови оптимального шляху з урахуванням статичних та випадкових перешкод, а також реалізовано механізм динамічного перепланування при виявленні перешкод або зупинок.

Під час експериментальної перевірки у середовищі Webots було підтверджено високу ефективність запропонованого підходу. Загальний

результат засвідчив успішний перехід від ручного до повністю автономного керування у середовищі з обмеженнями та перешкодами.

ВИСНОВКИ

У першому розділі було проведено огляд теоретичних основ автоматизованих систем керування рухомими об’єктами. Розглянуто історію розвитку, принципи побудови, класифікацію систем за архітектурою, рівнем автономності та сферою застосування. Детально проаналізовано основні алгоритми стабілізації (PID), планування (A*, RRT) і сучасні підходи на основі машинного навчання та цифрових двійників. Встановлено, що для побудови автономних систем ефективним є комбінування класичних і адаптивних методів із урахуванням особливостей середовища.

У другому розділі досліджено технічні аспекти реалізації автоматизованих систем. Проаналізовано архітектуру апаратної частини (сенсори, контролери, приводи) та програмного забезпечення (ОС реального часу, фреймворки, протоколи зв’язку). Показано, що ключовими вимогами до таких систем є надійність, адаптивність, здатність до обробки даних у реальному часі та модульність. Також окреслено перспективи використання цифрових двійників для підвищення точності й передбачуваності поведінки автономних об’єктів.

У третьому розділі розроблено і протестовано повноцінну систему автономного керування дроном. Реалізовано віртуальний прототип у Webots, впроваджено PID-регулятори для стабілізації, розроблено FSM-контролер та алгоритм навігації на основі A*. В результаті дрон успішно виконує зліт, переміщення між заданими точками, патрулювання, уникнення перешкод і повернення. Експериментальні результати засвідчили точність досягнення цілей ($\sigma \approx 0.9$ м), ефективність планування ($\approx$1.2 мс), стабільну роботу системи в складному середовищі.

У результаті дослідження було підтверджено, що поєднання класичних методів керування з алгоритмами планування маршруту в рамках єдиної програмної архітектури дозволяє досягти високого рівня автономності. Отримана система є адаптивною, стійкою до перешкод і здатна працювати в реальному часі з обмеженими ресурсами.

Результати цієї роботи будуть корисні для розробників автономних безпілотних систем, інженерів-робототехніків, дослідників у сфері керування рухомими об’єктами, а також студентів, що вивчають системи автоматизації, кіберфізичні системи чи інтелектуальні роботизовані платформи. Отримані підходи можуть бути адаптовані для мобільної робототехніки, автономного транспорту, дронів та систем індустрії 4.0.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вікімедіа, У. П. Відцентровий регулятор [Електронний ресурс]. – режим доступу: https://uk.wikipedia.org/wiki/Відцентровий_регулятор (дата звернення – 02.04.2025).
2. Вікімедіа, У. П. Автопілот [Електронний ресурс]. – режим доступу: https://uk.wikipedia.org/wiki/Автопілот (дата звернення – 15.05.2025).
3. Вікіпедія, У. П. Автоматизована система керування [Електронний ресурс]. – режим доступу: https://uk.wikipedia.org/wiki/Автоматизована_система_керування (дата звернення – 12.04.2025).
4. John Lewis. Motion Control Basics: The Engineering Behind Automation [Електронний ресурс]. – режим доступу: https://www.automate.org/motion-control/industry-insights/understanding-motion-control-basics (дата звернення – 18.04.2025).
5. Tikkanen & Amy. US Airways flight 1549 | Description, Pilot, & Facts [Електронний ресурс]. – режим доступу: http://britannica.com/topic/US-Airways-Flight-1549-incident (дата звернення – 03.05.2025).
6. TESLA. Офіційний сайт TESLA [Електронний ресурс]. – режим доступу: https://www.tesla.com/ (дата звернення – 20.04.2025).
7. Beasley Allen. Boeing 737 MAX 8 Crash | Aviation Accident lawsuit Updates [Електронний ресурс]. – режим доступу: https://www.beasleyallen.com/aviation-accidents/boeing-737-max-8-crashes/ (дата звернення – 28.04.2025).
8. Griggs, T., & Wakabayashi, D. How a Self-Driving uber killed a pedestrian in Arizona [Електронний ресурс]. – режим доступу: https://www.nytimes.com (дата звернення – 26.03.2025).
9. Вікіпедія. Пропорційно-інтегрально-диференціальний закон регулювання [Електронний ресурс]. – режим доступу: https://uk.wikipedia.org/wiki/Пропорційно-інтегрально-диференціальний_закон_регулювання (дата звернення – 10.04.2025).

10. GeeksforGeeks. Understanding Reinforcement Learning indepth [Електронний ресурс]. – режим доступу: https://www.geeksforgeeks.org/understanding-reinforcement-learning-in-depth/ (дата звернення – 25.04.2025).
11. Цивільний кодекс України. Стаття 1166. Загальні підстави відповідальності за завдану майнову шкоду [Електронний ресурс]. – режим доступу: https://kodeksy.com.ua/tsivil_nij_kodeks_ukraini/statja-1166.htm (дата звернення – 14.05.2025).
12. Марценко, Н. С., & Martsenko, N. Відповідальність за шкоду, завдану автономними транспортними засобами [Електронний ресурс]. – режим доступу: https://elartu.tntu.edu.ua/handle/lib/39006 (дата звернення – 06.05.2025).
13. StudySmarter. Automation Architecture: Role & Techniques [Електронний ресурс]. – режим доступу: https://www.studysmarter.co.uk/explanations/engineering/mechanical-engineering/automation-architecture/ (дата звернення – 07.04.2025).
14. Kutz, M. (Ed.). Instrumentation and Control Systems. – In Mechanical Engineers' Handbook (Vol. 3, pp. 3–45). John Wiley & Sons, 2013.
15. Siciliano, B., & Khatib, O. (Eds.). Springer Handbook of Robotics (2nd ed.). – Springer, 2016. – режим доступу: https://doi.org/10.1007/978-3-319-32552-1 (дата звернення – 01.05.2025).
16. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. ROS: An open-source Robot Operating System. – In ICRA workshop on open source software (Vol. 3, No. 3.2, pp. 1–6), 2009.
17. SimulInk. Simulation and Model-Based Design [Електронний ресурс]. – режим доступу: https://www.mathworks.com/products/simulink.html (дата звернення – 13.05.2025).
18. Bertocco, M., Ferraris, F., Offelli, C., Parvis, M., & Pozzobon, P. Data transmission in industrial applications: The impact of fieldbus systems. – IEEE Instrumentation & Measurement Magazine, 3(1), 20–30, 2000. – режим доступу: https://doi.org/10.1109/5289.824881 (дата звернення – 08.04.2025).

19. Tao, F., Sui, F., Liu, A., Qi, Q., Zhang, M., & Song, B. Digital twin-driven product design, manufacturing.
20. Cyberbotics. Robotics simulation with Webots [Електронний ресурс]. – режим доступу: https://cyberbotics.com/ (дата звернення – 23.03.2025).

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Бакалавр**
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

ВЛАСОВ Михайло Андрійович
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«СИСТЕМИ КЕРУВАННЯ АВТОМАТИЗАЦІЇ РУХОМИМИ ОБ'ЄКТАМИ»**
керівник роботи **Мірошник Марина Анатоліївна, професор кафедри, доктор технічних наук,**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від ***16 квітня 2025року №4101-5/962***

2. Строк подання студентом роботи ***30 травня 2025 року***

3. Перелік питань, які потрібно розробити:
   1. Дослідження основни методів та алгоритмів керування.
   2. Аналіз сучасних тенденцій, викликів і перспектив у сфері автономного керування.
   3. Вивчення архітектури автоматизованої системи.
   4. Формалізація задачі керування рухомими об'єктами та побудова математичних моделей.
   5. Аналіз алгоритмів планування траєкторії та оптимізації руху.
   6. Дослідження можливостей застосування МН у системах керування.
   7. Формування та підготовка навчального датасету.
   8. Реалізація алгоритму керування дроном із використанням нейромережі та reinforcement learning.
   9. Проведення симуляційного тестування розробленої системи.
   10. Оцінка ефективності системи за обраними метриками.
   11. Формулювання висновків і розробка пропозицій щодо покращення системи.

12. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Затвердження теми роботи та погодження з керівником	05.09.2024 - 09.09.2024
2	Збір літератури та аналітичних матеріалів щодо АСК рухомими об'єктами	10.09.2024 - 21.10.2024
3	Аналіз класифікації, типів та характеристик АСК	22.10.2024 - 18.11.2024
4	Дослідження алгоритмів керування: PID, MPC, DDPG	19.11.2024 - 21.12.2024
5	Аналіз архітектури систем керування та інтеграції з IoT / хмарними рішеннями	22.12.2024 - 02.12.2025
6	Розгляд методів машинного навчання у керуванні (RL, нейромережі)	03.12.2024- 01.02.2025
7	Підготовка датасету, попередня обробка даних, вибір програмного забезпечення	01.03.2025- 30.04.2025-
8	Реалізація алгоритму керування рухомим об'єктом	01.03.2025- 30.04.2025-
9	Тестування, оцінка результатів, побудова графіків ефективності)	01.03.2025- 30.04.2025-
10	Написання розділів пояснювальної записки, формування висновків	01.05.2025- 30.05.2025-
11	Оформлення пояснювальної записки та додатків відповідно до вимог	01.05.2025- 30.05.2025-
12	Аналіз результатів, формулювання висновків, підготовка до захисту	01.05.2025- 30.05.2025-
13	Представлення кваліфікаційної роботи керівнику та рецензенту	01.05.2025- 30.05.2025-

13. Дата видачі завдання ***02 жовтня 2024 року.***

Студент **Мірошник М. А.**

Керівник роботи **Власов М. А.**

Додаток Б

Затверджую

_____

«_____»_____2025 р.

Технічне завдання
на розробку програмного виробу «Автоматизовані системи керування рухомими об'єктами**»**

1.	Введення	1.1. Назва: Система автономного керування дроном у середовищі Webots з використанням алгоритму A* та PID-регулятора. 1.2. Галузь застосування: робототехніка, автоматизація, автономне управління, інтелектуальні системи керування.
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології 2.2. Завдання на кваліфікаційну роботу бакалавра № _____ від «___» _____2025 (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: створення, моделювання та експериментальна перевірка системи автономного керування дроном. 3.2. Призначення розробки забезпечити стабільну навігацію дрона за допомогою алгоритму планування шляху A*, стабілізаційного PID-регулятора та FSM-контролера у середовищі Webots. 3.3. Вихідні дані розробки: симуляційне середовище Webots, відкриті моделі дронів, власні сценарії тестування, алгоритми A*, PID.
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: рух дрона до цільової точки за планом A*, автоматичне уникнення перешкод, PID-стабілізація по висоті та осях, візуалізація маршруту та запис логів 4.2. Вимоги до надійності: точність стабілізації не гірша ніж ±1°, ефективність планування маршруту, стійкість до динамічних перешкод 4.3.Вимоги до умов експлуатації: середовище Windows або Linux, запуск в середовищі Webots 4.4. Вимоги до складу і параметрів технічних засобів: ПК середнього рівня, CPU не нижче Intel Core i5, оперативна пам’ять не менше 8 ГБ.

<table>
<tr><td></td><td></td><td colspan="2">4.5. Вимоги до інформаційної та програмної сумісності: інтерпретатор Python 3, підтримка Webots версії не нижче R2023b.
4.6. Вимоги до маркування та упаковки: немає
4.7. Вимоги до транспортування і зберігання: на звичайних носіях інформації
4.8. Спеціальні вимоги: немає.</td></tr>
<tr><td>5.</td><td>Вимоги до програмної документації</td><td colspan="2">Програмною документацією до виробу «Автоматизовані системи керування рухомими об'єктами» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику розрахунку планування (у вигляді розділу 3 пояснювальної записки до кваліфікаційної роботи).
3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)</td></tr>
<tr><td>6.</td><td>Вимоги до техніко-економічних показників</td><td colspan="2">Програмною документацією до виробу «Автоматизовані системи керування рухомими об'єктами» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику розрахунку планування (у вигляді розділу 3 пояснювальної записки до кваліфікаційної роботи).
3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)</td></tr>
<tr><td rowspan="5">7.</td><td rowspan="5">Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>до 15 жовтня 2024
до 25 листопада 2024</td><td>Аналіз практики предметної області та огляд алгоритмів A*, PID, FSM, архітектур дронів</td></tr>
<tr><td>від 2 грудня 2024
до 16 грудня 2024</td><td>Аналіз симуляційних платформ та програм Webots.</td></tr>
<tr><td>від 2 грудня 2024
до 2 січня 2025</td><td>Реалізація базової моделі дрона у Webots</td></tr>
<tr><td>від 3 січня 2025
до 30 березня 2025</td><td>Розробка та тестування PID-контролерів та FSM-архітектури</td></tr>
</table>

		від 11 лютого 2025 до 27 травня 2025	Реалізація алгоритму А*, моделлю дрона	інтеграц
		від 31 березня 2025 до 27 квітня 2025	Проведення експериментів та оцінка ефективності	
		від 1 травня 2025 до 28 травня 2025	Оформлення результатів. Написання пояснювальної записки.	
		30 травня 2025	Представлення кваліфікаційного проєкту керівнику кваліфікаційної роботи та рецензенту.	
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати в електронному вигляді в 1 примірнику.		

Виконавець
студент групи КУ- 41
Власов М. А.

Замовник
д.т.н., проф.
Мірошник М. А.

Додаток В

Програма і методика випробувань програмного виробу

«Автоматизовані системи керування рухомими об'єктами»

1 Об'єкт випробувань

1. Назва програмного виробу : «Система автономного керування дроном у середовищі Webots з використанням алгоритму A* та PID-регулятора»
2. Галузь застосування : автоматизація, мобільна робототехніка, моделювання автономних систем.
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

1. функціонувати на операційних системах Windows, Linux, MacOS,
2. забезпечувати стабільне позиціювання дрона з похибкою не більше ±1°,
3. містити захист від некоректного введення координат або станів,
4. бути сумісною з Webots, Python та бібліотеками контролю руху,
5. бути модульною для зручного розширення функціональності,

6. мати ізольовані підсистеми стабілізації, навігації, FSM-контролера,
7. запускатись на стандартному персональному комп'ютері (CPU ≥ i5, RAM ≥ 8 ГБ),
8. вимоги до маркування та упаковки (не висуваються);
9. вимоги до транспортування і зберігання (не висуваються).

Спеціальні вимоги (не пред'являються).

5. Вимоги до програмної документації

Програмною документацією до виробу «Система автономного керування дроном у середовищі Webots з використанням алгоритму A* та PID-регулятора» вважати:

1. Програмною документацією щодо розроблюваного програмного продукту вважати:
2. справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);
3. Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);
4. рекомендацій щодо застосування створеної програмної стандартизації у проєктах (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань необхіднио Webots, Python 3.9+, набір конфігурацій для запуску симуляцій дрона, логери, візуалізатор траєкторії, редактор сценаріїв польоту.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

-ознайомчий (1-й етап);

-випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірку комплектності програмної документації.
2. Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.
3. Перевірку комплектності складу технічних і програмних засобів.
4. Методику проведення перевірок на 1 етапі випробувань.
5. Якість програмної документації перевіряється на відповідність вимогам стандартів ЕСПД.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. перевірку відповідності технічних характеристик програми вимогам технічного завдання;
2. перевірку ступеня виконання функціональних вимог до програми;
3. методику проведення перевірок, що входять до переліку по 2 етапу випробувань.

1. Програма працює відповідно до умов експлуатації операційних систем MS Windows, Linux та MacOS.
2. Для роботи необхідний компілятор мови програмування python, версії не нижчої ніж 3.0
3. Порядок проведення випробувань:
   - 3.1. Запуск програми здійснюється шляхом запуску середовища Webots з попередньо завантаженим контролером дрона.
   - 3.2. У FSM-контролері активується одна з фаз: TAKEOFF, FLY_TO_POINT, RETURN_HOME, відповідно до обраного тесту.
   - 3.3. У процесі симуляції проводиться спостереження за поведінкою дрона у вікні моделювання, а також аналіз логів виконання у Webots Console.
   - 3.4. Результати випробувань фіксуються у вигляді графіків, анімацій маршруту та повідомлень у логах.

Для проведення випробувань пропонується тест 1, тест 2 та тест 3.

Тест 1

1. Перевірка виконання програми.
2. Активазація фази підняття у FSM-контролері.
3. Дрон виконує зліт до висоти 1 метра з подальшим зависанням.

Рис. В.1 Тест 1

Тест 2

1. Перевірка виконання програми
2. Планування траєкторії базовим алгоритмом A* FSM та з урахуванням статичних перешкод.
3. Дрон виконує політ.

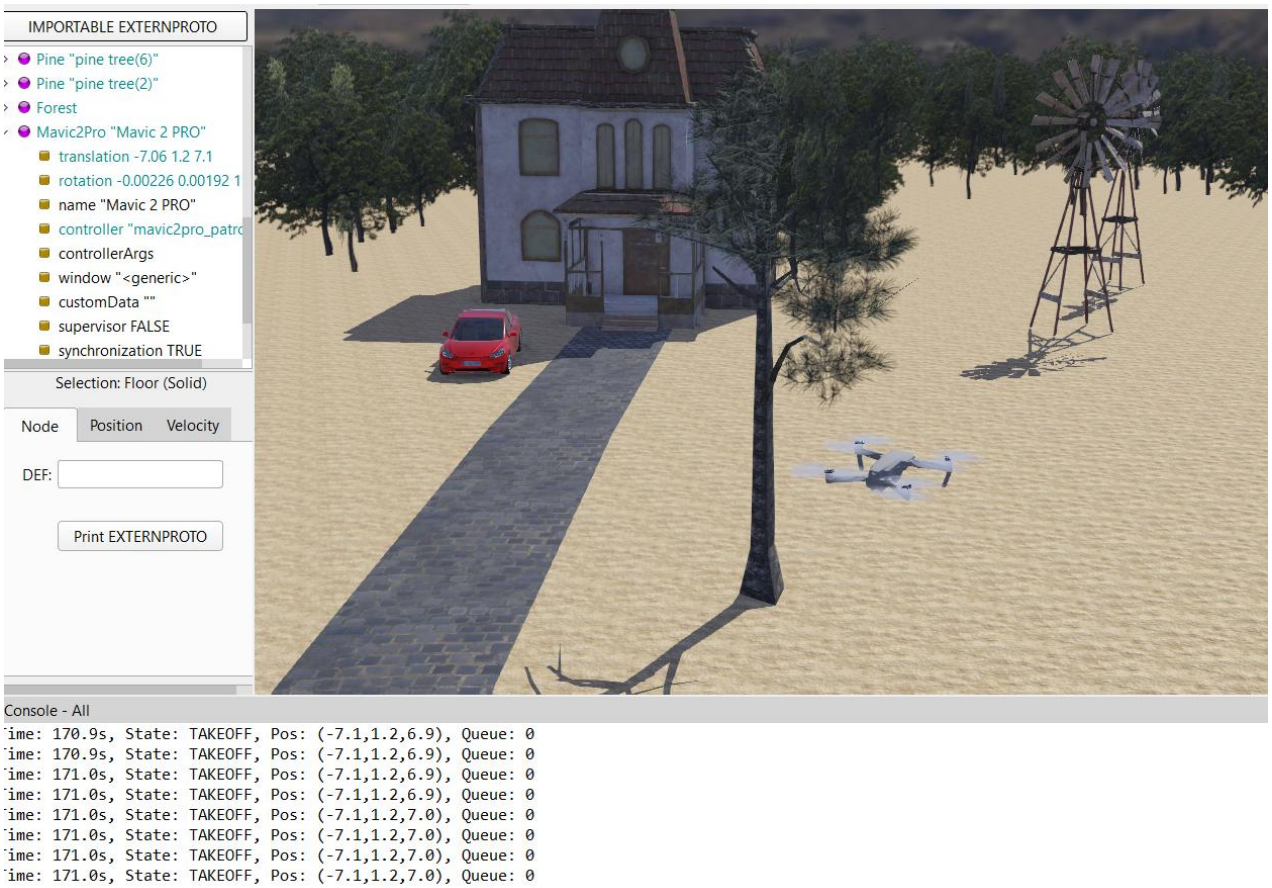

Рис. В.2 Тест 2

Тест 3

1. Перевірка виконання програми
2. Алгоритм A* повторно планує маршрут до точки зльоту з поточного положення
3. У логах Webots підтверджується завершення місії.

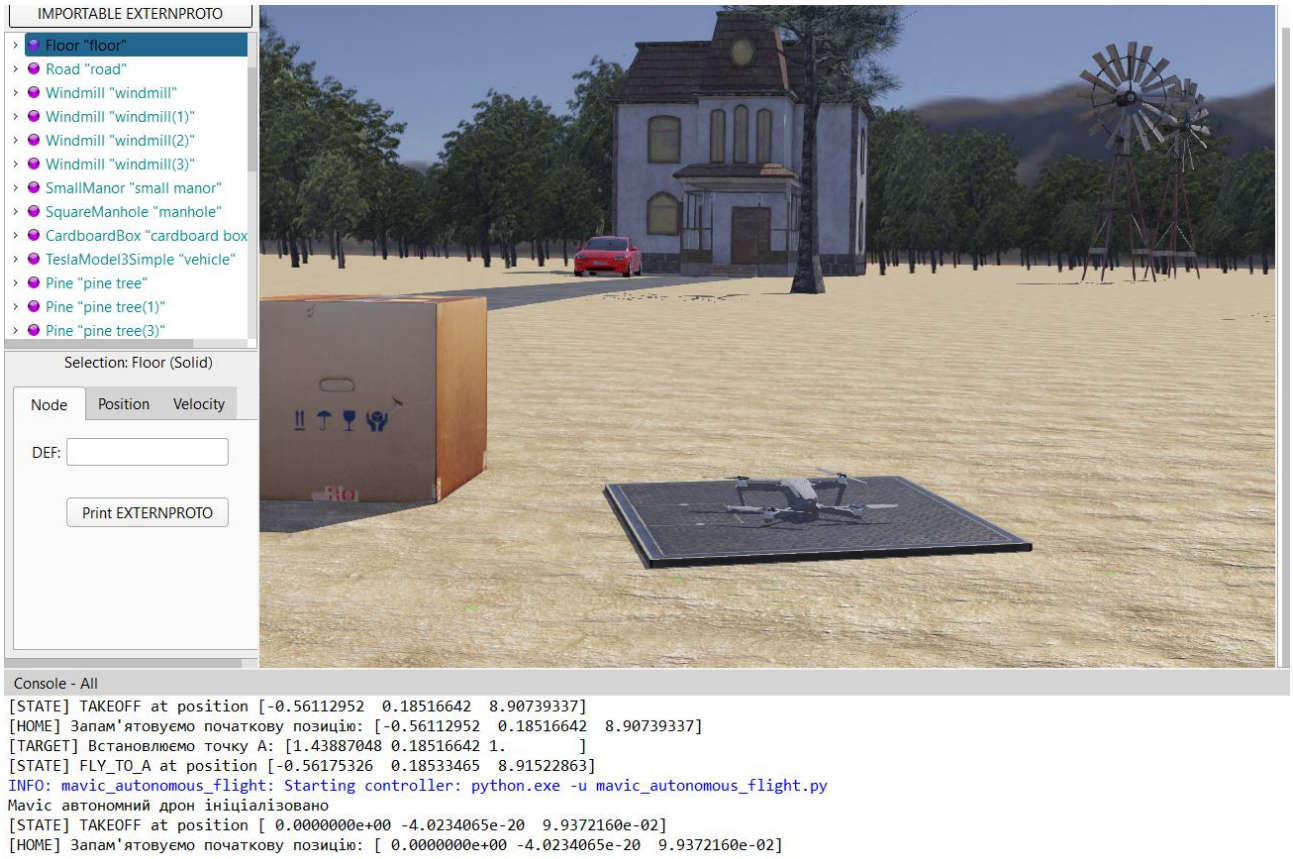

Рис. В.3 Тест 3

Виконавець: студент групи КІ-41, Власов М. А.