Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
## Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Вбудована система комбінованого нечіткого виведення в завданнях керування роботами

(тема)


Виконав:

студент II курсу, групи СПм-22-6

Олійник Д.Г.
(прізвище, ініціали)


Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: проф. Каргін А.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

# ЗАВДАННЯ

## НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Олійнику Дмитру Геннадійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Вбудована система комбінованого нечіткого виведення в завданнях керування роботами

затверджена наказом по університету від " 01 " квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи
Концепція AMR
Forward та Backward техніки виведення
Алгоритми пошуку у просторі станів, зокрема BFS
Мікроконтролерна система

4. Перелік питань, що потрібно опрацювати у роботі
1. Проблема автономності АІБС
2. Ціле-орієнтована модель управління
3. Підходи до нечіткого виведення
4. Підходи та методи вирішення завдань пошуку в просторі станів
5. Розробка алгоритму комбінованого зворотного виведення
6. Перевірка роботи алгоритму
7. Модель прототипу системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Лістинги алгоритмів, блок-схеми, схеми прототипу, 17 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз сучасного стану робототехніки | 02.04.24 – 12.04.24 | |
| 2 | Аналіз проблеми автономності АІБС | 12.04.24 – 22.04.24 | |
| 3 | Розробка алгоритму | 23.04.24 – 13.04.24 | |
| 4 | Тестування | 13.05.24 – 20.05.24 | |
| 5 | Опис моделі прототипу системи | 20.05.24 – 31.05.24 | |
| 6 | Оформлення матеріалів кваліфікаційної роботи | 01.06.23 – 08.06.23 | |
| 7 | Подання кваліфікаційної роботи керівникові та | 09.06.23 – 12.06.23 | |
| | її попередній захист | | |
| 8 | Подання роботи на рецензування | 12.06.23 – 15.06.23 | |
| | | | |
| | | | |
| | | | |

Дата видачі завдання 01 квітня 2024 р.

Студент _____ (підпис)

Керівник роботи _____ (підпис) проф. Каргін А.О. (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 67 с., 18 рис., 5 табл., 2 дод., 22 джерела.

АВТОНОМНІ РОБОТИ, БЕСПІЛОТНІ РОБОТИ, ВБУДОВАНІ СИСТЕМИ, ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ, НЕПЕРЕРВНЕ ПЛАНУВАННЯ, НЕЧІТКА ЛОГІКА, ПЛАНУВАННЯ ШЛЯХУ, РОБОТОТЕХНІКА, ШТУЧНИЙ ІНТЕЛЕКТ.

Метою кваліфікаційної роботи є розробка системи комбінованого нечіткого виведення у вбудовані системи, яка дозволить забезпечити автономність керування роботами в умовах невизначеності.

У ході виконання кваліфікаційної роботи було розроблено систему комбінованого нечіткого виведення для використання у вбудованих системах для вирішення завдання керування автономними роботами. Алгоритм передбачає наступне:

- прийняття у якості вхідних даних: базу знань у форматі правил «ЯКЩО-ТО», базу даних із відомими фактами, кінцевий факт як ціль пошуку;
- результатом роботи алгоритму є множина шляхів для розрахунку цільового факту з розрахованими факторами впевненості кожного шляху;
- відстеження знайдених проміжних фактів для зменшення кількості обчислень;
- уточнення ФВ шляхів у процесі їх доповнення знайденими необхідними фактами;
- відновлення шляху, якщо вже були знайдені відповідні правила.

Було описано апаратну та програмну модель прототипу системи, у якій зазначається опис алгоритму неперервного планування, шаблон програмного управління АІБС.

# ABSTRACT

Master’s thesis: 67 pages, 18 figures, 5 tables, 2 appendices, 22 sources.

AUTONOMOUS ROBOTS, UNMANNED ROBOTS, EMBEDDED SYSTEMS, INTELLIGENT SYSTEMS, CONTINUOUS PLANNING, FUZZY LOGIC, PATH PLANNING, ROBOTICS, ARTIFICIAL INTELLIGENCE.

The goal of this thesis is to develop a system of combined fuzzy inference for embedded systems that will ensure the autonomy of robot control under conditions of uncertainty.

During performing the work, a combined fuzzy inference system was developed for use in embedded systems in the task of controlling autonomous robots. The algorithm involves the following:

- accepting as input data: a knowledge base in the format of IF-THEN rules, a database with known facts, and the goal fact as the search target;
- tracking the found intermediate facts to reduce the number of calculations;
- adjusting the CF of the paths in the process of supplementing them with the necessary facts found;
- restoring the path if the corresponding rules have already been found;
- the result of the algorithm is a set of paths for calculating the goal fact with the calculated confidence factors of each path.

The hardware and software model of the system prototype is described, which includes a description of the continuous planning algorithm and the AIUS’s software control template.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АІБС – автономна інтелектуальна безпілотна система

БЗ – база знань

ГЗ – глобальні залежності

ЕЛС – нечітка логічна система

КЗ – комп’ютерний зір

КП – контекстна пам’ять

КФВ – комбінований фактор впевненості

РМ – розумна машина

ФВ – фактор впевненості

ШІВ – штучний інтелект, що відчуває

ШІМ – широтно-імпульсна модуляція

AGV – автоматизовані керовані транспортні засоби (англ. Automated Guided Vehicles)

AMR – автономні мобільні роботи (англ. Autonomous Mobile Robots)

AV – автономний транспортний засіб (англ. Autonomous vehicle)

BFS – пошук вширину (англ. Breadth-first search)

HRI – взаємодія між людиною та роботом (англ. Human-Robot interraction)

RC – дистанційне керування (англ. Remote control)

UART – універсальний асинхронний приймач-передавач (англ. Univsersal Asynchronos Reciever-Transmitter)

# ВСТУП

У сучасному світі робототехніка зазнає стрімкого розвитку та впровадження в різні сфери діяльності людини, від промисловості до побуту. Одним з ключових аспектів ефективного функціонування роботів є здатність здійснювати точне та надійне керування в умовах неповної або нечіткої інформації. Вбудовані системи комбінованого нечіткого виведення є перспективним напрямом в даній сфері, оскільки вони дозволяють поліпшити адаптивність і точність керування роботами в складних умовах.

На сьогоднішній день досягнуто значних успіхів у розробці вбудованих систем для роботів, які використовують нечітке логічне виведення. Такі системи успішно застосовуються для керування маніпуляторами, мобільними роботами та автономними транспортними засобами. Наприклад, системи нечіткого керування дозволяють роботам ефективно орієнтуватись в складних середовищах, уникати перешкод та виконувати точні маніпуляції з об'єктами.

Однак, незважаючи на значний прогрес, існують певні прогалини у знаннях та практичних напрацюваннях. Зокрема, залишається невирішеною проблема створення універсальних вбудованих систем, які б могли адаптуватися до широкого спектру задач і умов експлуатації. Багато існуючих рішень є специфічними для певних завдань і не можуть бути легко переналаштовані для інших умов. Крім того, актуальною є проблема зниження обчислювальної складності нечітких систем, що є критично важливим для вбудованих систем з обмеженими ресурсами.

У світовій практиці активно розробляються підходи до інтеграції комбінованих методів нечіткого виведення, таких як адаптивні нейронечіткі системи, генетичні алгоритми та методи машинного навчання. Ці підходи спрямовані на підвищення адаптивності та універсальності систем нечіткого керування. Крім того, ведуться дослідження у напрямку оптимізації

алгоритмів нечіткого виведення для зменшення обчислювальної складності та підвищення енергоефективності вбудованих систем.

Актуальність даної роботи обумовлена необхідністю створення ефективних і універсальних вбудованих систем керування для сучасних робототехнічних комплексів, які працюють в умовах невизначеності та зміни середовища. Розробка нових підходів до комбінованого нечіткого виведення дозволить значно покращити продуктивність та надійність робототехнічних систем, що, в свою чергу, сприятиме їх широкому впровадженню в різні сфери життя.

Метою даної роботи є розробка системи комбінованого нечіткого виведення у вбудовані системи, яка дозволить забезпечити автономність керування роботами в умовах невизначеності. Запропонована система буде орієнтована на використання в різних робототехнічних комплексах, включаючи промислові маніпулятори, мобільні роботи та автономні транспортні засоби.

Таким чином, результати цієї роботи матимуть широкий спектр застосувань, що сприятиме розвитку робототехніки та впровадженню передових технологій у практичну діяльність.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Огляд сучасного стану робототехніки

Робототехніка є комплексною галуззю, яка інтегрує комп'ютерні науки та інженерію з метою автоматизації, полегшення та підтримки взаємодії людини з реальним світом. Вона охоплює широкий спектр технологій, де обчислювальний інтелект інтегрується у фізичні машини, що створює системи з можливостями, які значно перевершують потенціал окремих компонентів. Такі системи здатні виконувати завдання, недосяжні для звичайних машин або навіть для людей з традиційними інструментами. Автономність, або здатність машин самостійно рухатися і виконувати завдання, без людського втручання, є однією з ключових можливостей, що відкриває численні унікальні сфери застосування роботизованих систем [2, 3].

Робот – це механізм, який заздалегідь програмується для виконання складних серій дій із певним ступенем автономності, метою яких є виконання переміщень, маніпуляцій, позиціонування, збору даних тощо [4, 5].

Роботи мають здатність виконувати різноманітні завдання, що зазвичай виконуються людьми, включаючи ті, які вимагають великої дбайливості та складності. Вони виявляються значно більш точними та ефективними у деяких аспектах, зокрема у сферах виробництва масового масштабу. Окрім цього, роботи застосовуються у ситуаціях, де існує потенційна загроза для життя людини, таких як виявлення та розмінювання вибухових пристроїв, очищення небезпечних речовин та виконання завдань у надзвичайних умовах: під водою, у космосі, при високих температурах тощо [3].

Класифікації роботів є нетривіальною задачею, адже в залежності від сфери їх використання, типів задач вимоги будуть змінюватися. Наприклад, для роботів, що використовуються у сфері сільського господарства використовують наступні категорії для класифікації [6, 7]:

- тип роботи/функції (обробка полів, удобрення, картографування, моніторинг, маніпулювання об’єктами тощо);
- мобільність (стаціонарні, мобільні);
- середа пересування (повітря, земля, вода);
- автономність (керування людиною-оператором, напівавтономний, автономний).

Тому для загального розуміння сучасного стану робототехніки доцільніше розглядати більш широку та узагальнену класифікацію [4, 8], яку наведено на рисунку 1.1.

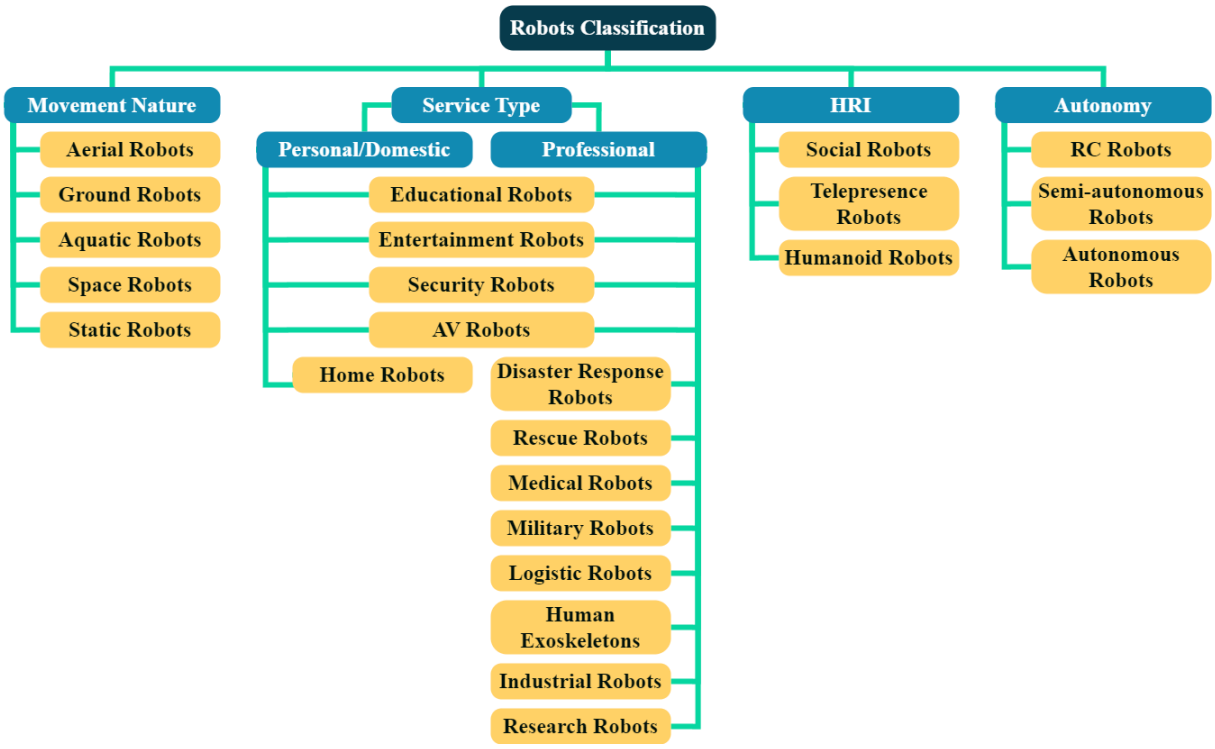
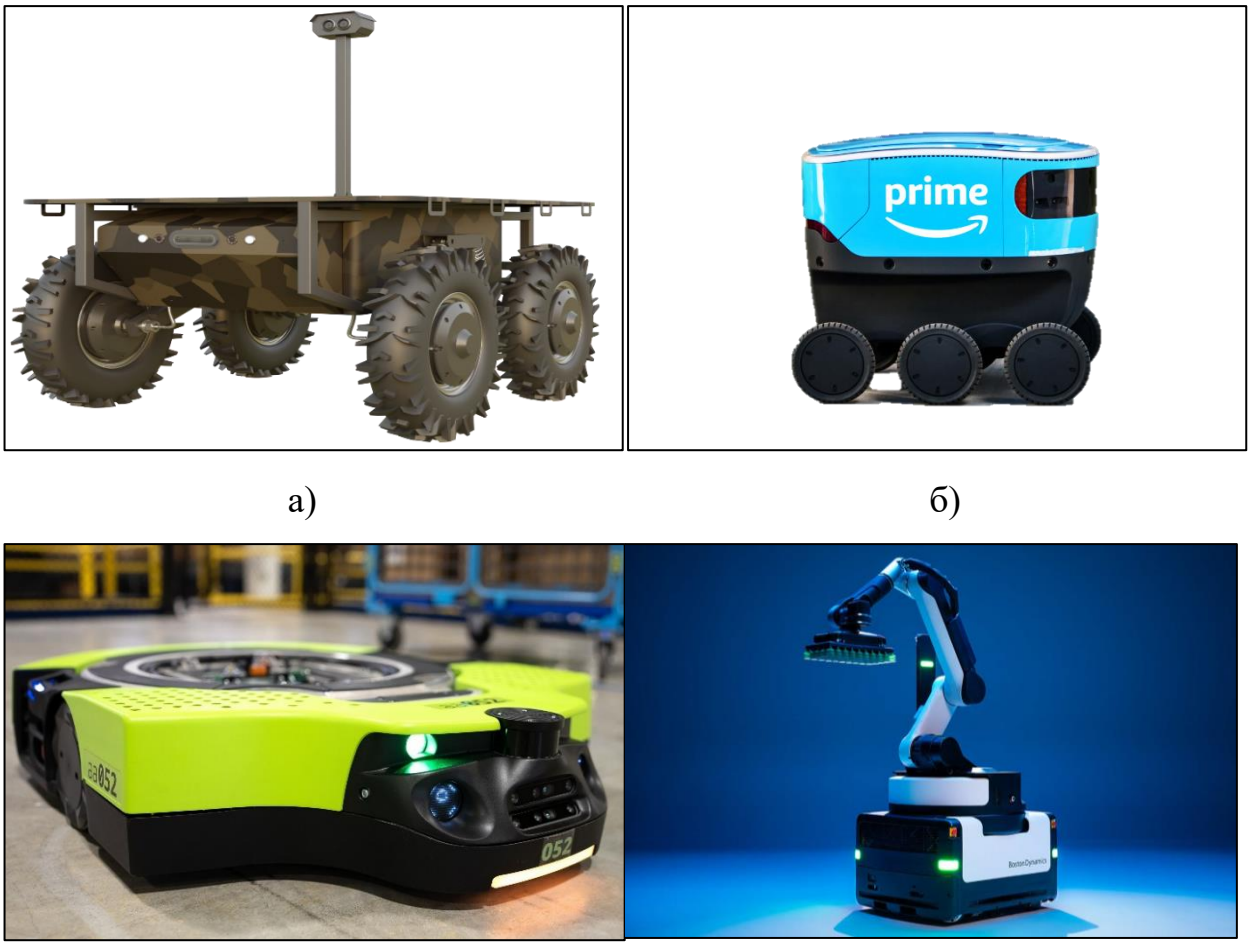



Рисунок 1.1 – Сучасна загальна класифікація роботів

Логістичні роботи – це широкий клас роботів, які використовуються для виконання логістичних задач. Це можуть бути як і роботи-кур’єри, що використовуються за межами будівель, приміщень, та які за допомогою своїх навігаційних систем та використання технології КЗ мають змогу доставляти

вантажі з відправної точки до місця призначення у різних умовах навколишньої середи (рисунок 1.2 а, б).

Однак це також можуть бути і роботи, що використовуються в межі приміщення або будівлі (переважно у складських приміщення). Такі роботи зазвичай виконують завдання переміщення вантажу, його сортування, складання (рисунок 1.2 в, г).




а) б)

в) г)

Рисунок 1.2 – Логістичні роботи: а) логістично-евакуаційна платформа «Тайра», б) Amazon Scout, в) Amazon Proteus, г) Boston Dynamics Stretch

Таким чином, логістичний робот – це самостійний пристрій або компонент системи, завданням якого є взаємодія та переміщення вантажу. Для виконання завдань такого типу вони можуть мати наступний функціонал:

- часткова або повна автономність;
- навігація (комплекс програмно-аппаратних засобів для орієнтації у навколишньому середовиші);
- маніпулювання об’єктами (роботизована рука з клішнею, присосками);
- можливість сприйняття, як частина системи навігації, автономності; планування шляху.

Поточними проблемами та напрямами сучасних досліджень робототехніки в контексті логістичних роботів є [4]:

- продовження розвитоку AGV та AMR;
- розвиток повної автономності роботів, особливо питаннях у прийняття рішень;
- інтелектуальне сприйняття навколишнього середовища;
- мережево-центриність.

Популярним класом роботів на сьогодні також є промислові роботи для сільського господарства. Цей клас налічує велику кількість різноманітних типів роботів [9]:

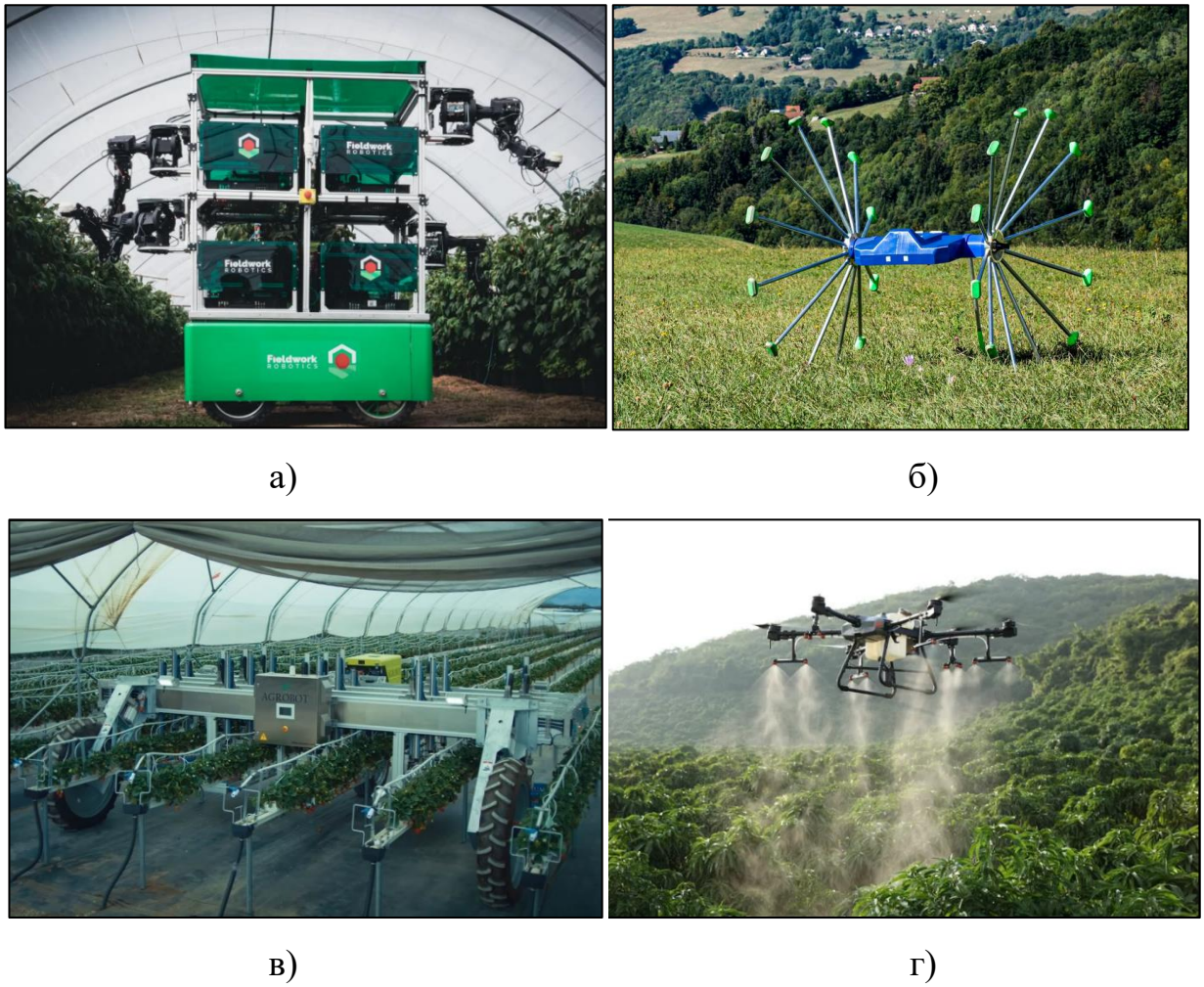
- роботи, що поливають;
- роботи-садівники;
- роботи, що удобрюють (рисунок 1.3 г);
- роботи-інспектори (рисунок 1.3 б);
- роботи-збирачі (рисунок 1.3 а, в);
- робот-робітник польовий.

Ці типи охоплюють широкий спектр завдань, які спрямовані на сільськогосподарську діяльність. Представники таких роботів переважно є наземними, проте для певного роду завдань використовуються також і повітряні дрони.

Типовими функціями для цього класу роботів є:

- різний ступінь автономністі (від повністю ручного керування до повністю автономного);

- навігація (комплекс програмно-аппаратних засобів для орієнтації у навколишньому середовиші);
- маніпулювання об’єктами (роботизована рука з клішнею);
- аналітичні засоби (КЗ).




а) б)

в) г)

Рисунок 1.3 – Сільськогосподарські роботи: а) робот-збирач Fieldwork Robotics Kraken-like robot, б) робот-інспектор Meropy SentiV, в) робот-збирач Agrobot E-Series, г) робот-розпилювач DJI Agras T-30

Поточними проблемами та напрямами сучасних досліджень робототехніки в контексті сільськогосподарських роботів є [4, 10]:

- розвиток повної автономності роботів;
- навігація;

- впровадження мережево-центриності;
- інтелектуальне сприйняття навколишнього середовища.

Таким чином можна бачити, що робототехніка є актуальною галуззю інженерії, що стрімко розвивається, а її результати, безпосередньо роботи – знаходять застосування у широкому спектрі людської діяльності: від виконання рутинних домашніх завдань (наприклад робот-пилосос) до прийняття участі у ліквідації лих, розмінуванні та космічних операціях.

Серед сучасних проблем та тенденцій розвитку робототехніки простежуються наступні:

- підвищення автономності роботів;
- надання можливості інтелектуального сприйняття навколишнього середовища;
- покращення аналітичних можливостей роботів;
- впровадження мередево-центричності.

В кваліфікаційній роботі пропонується розглянути проблему автономності роботів.

## 1.2 Проблема автономності АІБС на прикладі колісного роботу

Нехай є АІБС, завданням якої є постійне перевезення певного вантажу роботом з позиції А на позицію В (рисунок 1.4). Для успішного виконання завдання з перевезення вантажу роботом з позиції А на позицію В і назад, робот має виконати певні етапи. Ці етапи можна розглядати як локальні цілі (1.1), які разом формують план реалізації місії, який є послідовністю станів системи «Оточення-АІБС». Наведемо приклад послідовності станів/етапів, що відображає цей план:

- завантаження вантажу на позиції А: АІБС переміщується до позиції А та завантажує вантаж.
- переміщення АІБС на позицію 1;
- переміщення АІБС на позицію 4;

- переміщення АІБС на позицію 5;
- переміщення АІБС на позицію В;
- розвантаження вантажу АІБС на позиції В: АІБС переміщується до позиції В та розвантажує вантаж;
- переміщення АІБС на позицію 5;
- переміщення АІБС на позицію 3;
- переміщення АІБС на позицію 2;
- переміщення АІБС на позицію 1;
- переміщення АІБС на позицію А.

Далі описана послідовність станів/етапів повторюється. Таким чином, циклічне повторення процесу забезпечує безперервне перевезення вантажу між позиціями А та В (рисунок 1.4).

$$\rightarrow A^{Load} \rightarrow 1^{Move} \rightarrow 4^{Move} \rightarrow 5^{Move} \rightarrow B^{Unload} \rightarrow 5^{Move} \rightarrow 4^{Move} \rightarrow 1^{Move} \rightarrow A^{Load} \rightarrow . \quad (1.1)$$

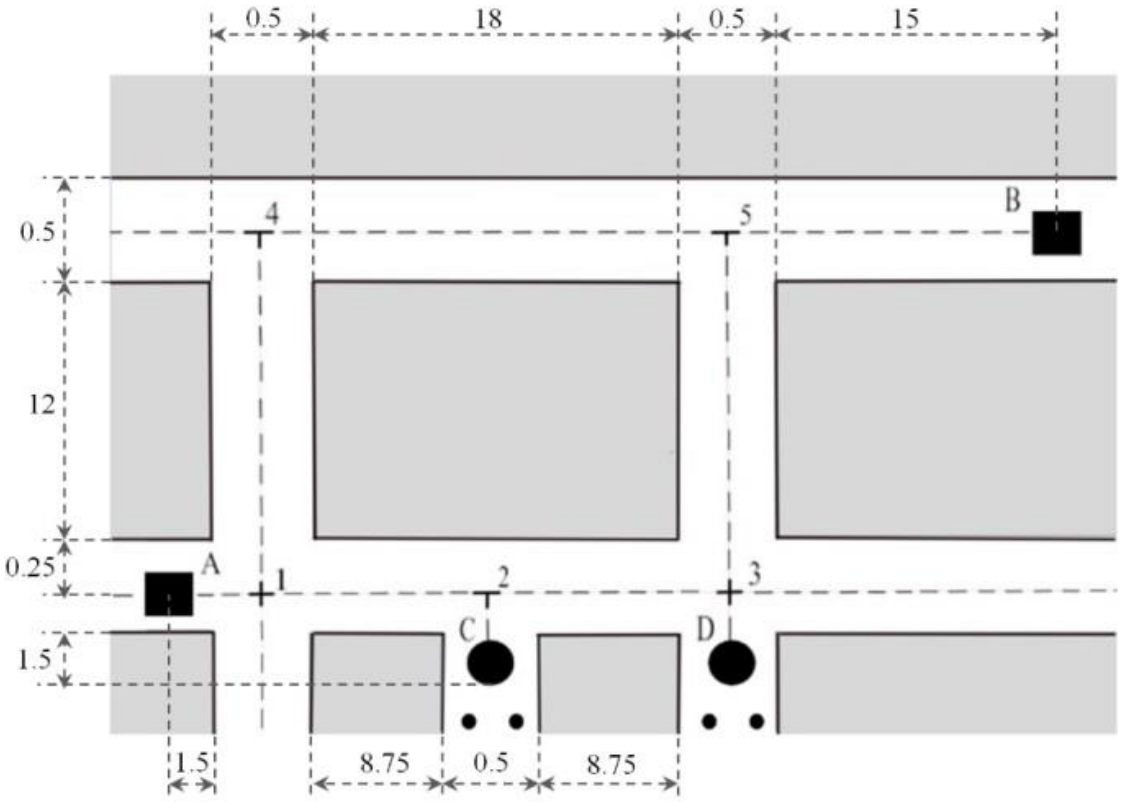



Рисунок 1.4 – Модель оточення АІБС

Важливо розуміти, що виконання локальної цілі може являти собою складну інкапсульовану послідовність дій, тому вона може бути піддана процесу декомпозиції для представлення кожної локальної цілі у формі більш деталізованого, атомарного плану дій. Також важливо враховувати, що АІБС використовують концепцію AMR замість AGV для навігації, що дозволяє підвищити рівень автономності та розширити можливості робота. Це дозволяє збільшити межі місії робота від переміщення виключно за фіксованим шляхом до подолання можливих перешкод. Тому план реалізації місії має бути гнучким, що передбачає: додавання етапів, для обходу перешкод, до плану завдання; коригування, або повну перебудову плану за потреби; сприяння досягненню кінцевої мети. До можливих перешкод можна зарахувати пошкодження оточення, на яке спирається АІБС (мітки, розмітка тощо), поява об’єктів, що перешкоджають рух, або нестача заряду батареї робота.

Керування реалізацією плану дій в умовах невизначеності, що змінюються, для досягнення кінцевої мети є завданням ШІВ [11; 12]. Проте, головним недоліком нечітких логічних систем для виконання завдань ситуаційного управління в комбінації з безперервним плануванням – є відсутність можливості утримання контексту, що призводить до відсутності повного розуміння поточного стану системи і необхідних дій на кожному етапі плану для виконання місії. Випадковий вибір траєкторії дій не сприяє ефективному виконанню плану місії, адже послідовність станів системи вимагає руху за певною траєкторією. Це обмеження нечітких систем пропонується подолати шляхом введення контекстної залежності в множину незалежних нечітких правил БЗ [11], що дозволяє додати послідовність локальних цілей етапів плану, зберігаючи незалежність правил.

### 1.2.1 Ціле-орієнтована модель управління, що залежить від контексту

Для впровадження контекстно залежної моделі керування до структури нечіткої системи потрібно зробити ряд змін, так само як і до БЗ та фактів [11].

Для цього потрібно розділити множину фактів F на дві підмножини, де перша містить факти $F^{plan}$, на яких визначають етапи плану (локальні цілі), а друга – факти $F^{sit}$ стосовно ситуації, в якій перебуває оточення АІБС.

$$F = \{F^{plan} = \{f_i, i = 1,2,\dots,n\} = \\ = \{A^{Load}, B^{Unload}, 1^{Move}, 2^{Move}, 3^{Move}, 4^{Move}, 5^{Move}\}, F^{sit} = \{g_j, j = \\ = 1,2,\dots,m\}\}. \quad (1.2)$$

Далі створюється відображення кожного факту локальної цілі з $F^{plan}$ (1.2), яке вводиться у КП у вигляді відповідного додаткового контекстного факту задля збереження стадій виконання плану.

Відповідно до кожного факту $f_i$ з (1.2), який надійшов від сенсорів, позначимо цей контекстний факт як $^*f_i$. Таким чином, фактам за $F^{plan}$ відповідатимуть наведені в КП контекстні факти з множини F.

$$F = \{^*A^{Load}, ^*B^{Load}, ^*1^{Move}, ^*2^{Move}, ^*3^{Move}, ^*4^{Move}, ^*5^{Move}\}. \quad (1.3)$$

Стан НЛС на довільному етапі реалізації плану визначає вектор *CF* значень ФВ у базі фактів:

$$CF = (cf(f_i), cf(^*f_i), i = 1,2,\dots,n, cf(g_j), j = 1,2,\dots,m). \quad (1.4)$$

Процес виконання плану місії разом із правилами НЛС, які стосуються шару управління для реалізації плану, супроводжується оновленням значень ФВ фактів у КП, які є елементами підмножин $F^{plan}$ та $F^{sit}$, базуючись на даних реального світу від сенсорів, що встановлені на АІБС.

Знання АІБС це – досвід у вигляді фрагментів траєкторії поведінки, що призвела до досягнення мети при виконанні місії або усунення перешкод. Ці фрагменти являють собою, серед іншого, різні варіанти реалізації планів дій. Кожен фрагмент представляє структуру знань у вигляді рисунку 1.5.

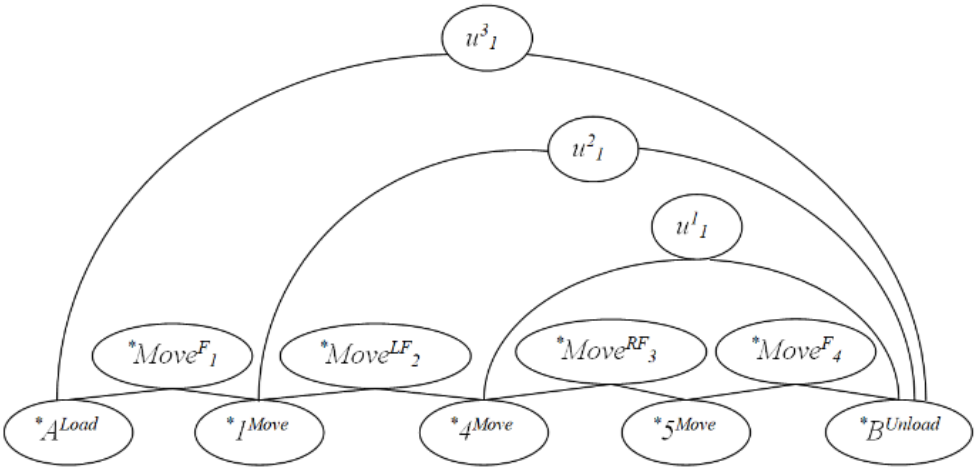



Рисунок 1.5 – Структура частини знань бази знань, що відповідає правилам $R_1$-$R_7$

На нижньому рівні структури це є елементарні знання що даються у формі «факт$_1$-дія-факт$_2$», які відображають, що за наявності факту$_1$ акт дії призводить до появи факту$_2$. В (Б.1) наведено набір правил, які представляють фрагмент реалізації місії в оточенні (рисунок 1.4), що потребується для завантаження вантажу в точці А та розвантаженні в точці В. Структура відображає не тільки елементарні причинно-наслідкові зв'язки, а й стійкі більш тривалі ланцюги зв'язків. Наприклад, мета $^*B^{Unload}$ досягається (можна отримати факт $^*B^{Unload}$), якщо за наявності факту $^*A^{Load}$ реалізується дія $u^3_1$, яка є узагальненням послідовності дій $^*Move^F_1$ і $u^2_1$, де верхній індекс визначає характер руху (F – рухатися прямо, LF – повернути наліво та рухатись прямо, RF – повернути направо та рухатись прямо), а нижній індекс є ідентифікаційним номером дії. Стосовно позначок дій високого рівня $u^l_n$ – l позначає рівень ієрархії правила, n – номер сценарію руху, що реалізується. Так само мета $^*B^{Unload}$ досягається, якщо за наявності факту $^*1^{Move}$ реалізується дія $^*u^2_1$, яка є узагальненням послідовності дій $^*4^{Move}$ ТА $u^1_1$. В свою чергу, $u^1_1$ є уособленням послідовності $^*Move^{RF}_3$ та $^*Move^F_4$. Ці знання представлені в Базі Знань (БЗ) правилами $R_5$, $R_6$ і $R_7$, відповідно.

Оскільки рух реалізується за умови наявності перешкод, в якості прикладу, наведемо одну з них, що є нестачею заряду батареї робота. АІБС має

вирушити до зарядної станції C або D, і лише потім повернутися до виконання плану.

У випадку жорсткої нестачі енергії обирається маршрут 2, за яким АІБС рухається до точки C для виконання зарядки з виконанням узагальнюючих дій $u^2_2$ та $u^2_1$ (правила $R_8$-$R_{12}$). Після активації зарядки далі може бути реалізований маршрут 4, за яким робот може досягти кінцевої дії ${}^*B^{Unload}$ (правила $R_{20}$-$R_{26}$). Подібний сценарій також реалізовано для випадку помірної нестачі енергії, коли АІБС спочатку рухається до точки D для виконання зарядки, і вже потім до мети ${}^*B^{Unload}$.

Ціле-орієнтований механізм ШІВ реалізує нечітке логічне виведення з комбінацією методів прямого та зворотного ланцюгового висновку з ФВ. Прямий ланцюговий висновок використовує ФВ з бази фактів $F^{plan}$ та $F^{sit}$ а зворотній ланцюговий висновок використовує факти з КП. Перед активацією механізму виведення ФВ фактів з $F^{plan}$ та $F^{sit}$ описують реальний стан середовища та етапів плану. Факти з КП відображають стани, що характеризують можливість реалізації дій, що ведуть до досягнення локальних цілей. Наприклад, $cf^{Re} = 0,9$ факту ${}^*Move^{F}_4$ відображає можливість реалізації переходу від ${}^*5^{Move}$ до ${}^*B^{Unload}$. Інше значення $cf^{Re} = –0,9$ свідчить про неможливість досягнення локальної мети ${}^*A^{Load}$ таким шляхом (правило $R_4$ в (Б.1)). Ці значення змінюються відповідно до змін стану навколишнього середовища в реальному часі.

## 1.3 Огляд підходів до нечіткого виведення

### 1.3.1 Forward-chaining

Вирішення деяких проблем природно починається зі збору інформації, на основі якої застосовується логічне виведення для отримання висновків. Наприклад, банк починає процес оцінки заяви на кредит, запитуючи у заявника про його фінансовий стан, такі як доходи, кредитна історія, наявність

інших кредитів тощо. Потім банк використовує цю інформацію для формулювання гіпотези або висновку щодо надання кредиту. Цей спосіб логічного виведення в експертних системах називається прямим виведенням.

Пряме виведення є обчислювальною моделлю знизу вгору. Воно починається з набору відомих фактів і застосовує правила для генерування нових фактів, чиї умови відповідають відомим фактам. Цей процес триває, поки не буде досягнута задана мета або поки не вичерпаються нові факти, які можна вивести. Пряме виведення перевіряє факти відносно запиту або мети, що вказує на те, що виведення рухається від фактів до мети. Воно також використовується для вдосконалення експертних систем та моделювання роботи людського мозку в штучному інтелекті [13].

Застосування прямого виведення в експертній системі на основі правил виглядає наступним чином:

- користувач надає системі інформацію про проблему;
- ця інформація зберігається в БД;
- інференційний двигун сканує БЗ в заданій послідовності, шукаючи ті, чиї умови відповідають вмісту БД. Якщо знаходиться таке правило, його висновок додається до БД;
- система знову перевіряє БЗ, шукаючи нові відповідності;
- на новому циклі правила, які вже спрацювали, ігноруються;
- ця процедура триває, поки не буде знайдено нових відповідностей;
- у БД міститься інформація, надана користувачем, та висновки системи.

Припустимо, що людина подала заявку на кредит, і її дані наступні: вона має стабільний дохід, хорошу кредитну історію, але відсутній власний капітал. Спочатку інформація про заявника вводиться в систему і порівнюється з умовами першого правила. Система виявляє, що немає відповідності, оскільки заявник не має власного капіталу. Тому висновок першого правила (відхилити заявку) не розглядається. При перевірці другого правила виявляється, що умови Правила 2 відповідають інформації про заявника, тому висновок

(надати кредит з підвищеною процентною ставкою) додається до робочої пам'яті. Далі перевіряється Правило 3, але відповідності з інформацією про заявника немає, оскільки заявник не має відмінного кредитного рейтингу і високого доходу. Система перевіряє всі правила, доки не залишиться жодного, яке ще не перевірялося. У робочій пам'яті залишається порада: надати кредит з підвищеною процентною ставкою.

### 1.3.2 Backward-chaining

Процес зворотного виведення схожий на тестування гіпотез у людському вирішенні проблем. Наприклад, банк може підозрювати, що клієнт має високий ризик не платежу, і намагається підтвердити це, шукаючи відповідні фінансові дані. Такий спосіб логічного виведення моделюється в експертній системі за допомогою пошуку, орієнтованого на мету, і називається зворотним виведенням. Це обчислювальний дизайн зверху вниз, що починається з мети або гіпотези і шукає правила, які підтримують цю гіпотезу. Зворотне виведення намагається знайти факти, які задовольняють мету, рухаючись від мети до фактів [13].

Застосування зворотного виведення в експертній системі на основі правил виглядає наступним чином:

- система перевіряє БД, щоб визначити, чи була мета вже досягнута. Інша БЗ могла вже підтвердити мету, тому цей крок є необхідним. Система шукає правила, які містять мету в частині THEN;
- система перевіряє, чи перераховані умови правил мети в БД. Якщо умови не перераховані, вони стають новими цілями, які мають бути доведені і можуть підтримуватися іншими правилами;
- система продовжує цей рекурсивний процес, поки не знайде умову, яка не підтримується жодним правилом (примітив). Коли знаходиться примітив, система запитує у користувача інформацію про нього. Система використовує цю інформацію для підтвердження підцілей та початкової мети.

## 1.4 Огляд підходів і методів вирішення завдань пошуку в просторі станів

Пошук у просторі станів є фундаментальною концепцією у завданнях вирішення проблем, у штучному інтелекті, що використовується у різних сферах для дослідження потенційних рішень заданої проблеми.

Пошук у просторі станів – це процес систематичного дослідження простору можливих станів або конфігурацій в області розв'язання проблеми з метою пошуку рішень або оптимальних шляхів. Він знаходить широке застосування у штучному інтелекті, робототехніці, дослідженні операцій та інших галузях, де проблеми можна моделювати як набір станів і переходів між ними. Значення пошуку в просторі станів полягає в його здатності ефективно орієнтуватися в складних проблемних просторах, що дозволяє знаходити рішення навіть у великих комбінаторних просторах.

Проблеми часто моделюються у вигляді простору станів, набору станів, в яких може перебувати проблема. Множина станів утворює граф, де два стани з'єднуються, якщо існує операція, яку можна виконати для перетворення першого стану в другий.

Пошук у просторі станів часто відрізняється від традиційних методів пошуку в інформатиці, оскільки простір станів є неявним: типовий граф простору станів занадто великий, щоб генерувати і зберігати його в пам'яті. Замість цього, вузли генеруються в міру їх дослідження, а потім, як правило, відкидаються. Розв'язанням комбінаторного пошуку може бути сам цільовий стан або шлях від деякого початкового стану до цільового.

Властивостями будь-якого методу пошуку в просторі станів є:

- повнота: пошук вважається повним, якщо він гарантує повернення принаймні одного розв'язку, якщо такий існує, для будь-якого випадкового входу;
- оптимальність: якщо завдяки алгоритму було знайдено рішення, що гарантовано є кращим (наприклад найнижча вартість шляху) серед інших, то таке рішення є оптимальним;

- часова складність: оцінка часової тривалості виконання алгоритму;
- просторова складність: максимальний об’єм пам’яті, який необхідний у будь-який момент часу виконання пошуку.

Підходи і методи вирішення завдань пошуку в просторі станів можна розділити на дві групи: неінформовані, інформовані.

### 1.4.1 Методи неінформованого пошуку

Методи неінформованого пошуку – це клас пошукових стратегій загального призначення, які не враховують місцезнаходження цілі пошуку [14], та які працюють за принципом грубої сили. До таких методів можна віднести:

- пошук в ширину;
- пошук в глибину;
- пошук в глибину з ітеративним заглибленням;
- алгоритм Дейкстри;
- двонаправлений пошук.

Особливістю такої групи методів є повнота пошуку, при якій, якщо існує краще рішення – воно буде знайдено. Запропонований алгоритм у лістингах 2.1-2.4 був натхненний алгоритмом пошуку в ширину. У лістингу 1.1 наведено приклад алгоритму пошуку в ширину.

Лістинг 1.1 – Приклад алгоритму пошуку в ширину

```
   Input connected graph G that consists of vertexes; starting node start; goal node target.
   Output index of target node; distance from start to target.
start BFS
Step 1. Primary initialization
   for each v from vertexes do
      set explored[v] to false
      set distance[v] to ∞
   end for each
   set explored[start] to true
   set distance[start] to 0
```

```
   put start to queue
Step 2. Exploration of nodes
   while queue is not empty do
      pop current from queue
      for each v adjacent to current do
         if not explored[v] then
            set explored[v] to true
            set distance[v] to distance[current] + 1
            put v to queue
            if v is target then
               return (v, distance[v])
            end if
         end if
      end for each
   end while
   return (-1, ∞)
end BFS
```

### 1.4.2 Методи інформованого пошуку

Методи інформованого пошуку – це клас пошукових стратегій загального призначення, які враховують додаткову інформацію про кожен наступний можливий стан для обрання найбільш перспективного [14], як: як далеко агент знаходиться від мети; вартість шляху; як дістатися до цільового вузла. Ці знання допомагають менше досліджувати простір пошуку та ефективніше знаходити цільовий вузол.

Алгоритм інформованого пошуку більш корисний для великого простору пошуку. Алгоритми інформованого пошуку використовують ідею евристики, тому їх ще називають алгоритмами евристичного пошуку.

Даний клас алгоритмів базується на евристичній функції. Евристична функція – це функція, яка використовується в інформованому пошуку і знаходить найперспективніший шлях. Вона приймає поточний стан агенту як вхідні дані і видає оцінку того, наскільки близько агент знаходиться до мети. Евристичний метод, однак, не завжди може дати найкраще рішення, але він гарантовано знайде задовільне рішення за прийнятний час. Евристична функція оцінює, наскільки близьким є стан до мети. До таких методів можна віднести:

- жадібний алгоритм;
- алгоритм пошуку A*;
- метод гілок і меж.

## 1.5 Постановка задачі

Завданням кваліфікаційної роботи є розробка системи комбінованого нечіткого виведення.

Для виконання завдання необхідно, щоб було виконано наступні пункти:

- розробити алгоритм комбінованого зворотного виводу;
- описати апаратну та програмну модель прототипу системи.

Алгоритм має передбачати наступне:

- алгоритм повинен приймати у якості вхідних даних: базу знань у форматі правил «ЯКЩО-ТО», базу даних із відомими фактами, кінцевий факт як ціль пошуку;
- алгоритм має відстежувати знайдені проміжні факти для зменшення кількості обчислень;
- алгоритм має уточнювати фактори впевненості шляхів у процесі їх доповнення знайденими необхідними фактами;
- алгоритм повинен мати можливість відновлення шляху, якщо вже були знайдені відповідні правила;
- результатом роботи алгоритму має бути множина шляхів для розрахунку цільового факту із вже відомими факторами впевненості кожного шляху.

Опис апаратної та програмної моделі прототипу системи має враховувати наступні аспекти: опис компонентів апаратної частини, опис архітектури апартної частини, опис архітектури та компонентів програмної частини, алгоритм неперервного планування, абстрактний шаблон програмного управління АІБС.

# 2 РОЗРОБКА АЛГОРИТМУ

## 2.1 Опис алгоритму комбінованого зворотного виводу

Алгоритм представляє собою інтерпретацію методу зворотного виводу, який дозволяє отримати усі можливі шляхи розрахунку заданого цільового факту з розрахованими ФВ кожного знайденого шляху та отримати КФВ.

Алгоритм базується на використанні ідеї алгоритму пошуку в ширину та створенні дерева шляхів розрахунку цільового факту. Кожен вузол дерева представляє собою правило, яке необхідно розрахувати, щоб отримати цільовий факт або один з фактів, що необхідні для його розрахунку. Структура дерева має стандартний вигляд: дерево має корінь, гілки, листя. Однак, структура вузла дерева була модифікована для можливості реалізації алгоритму.

Формальне визначення вузла дерева має вигляд:

$$< N, R_N, P_N, \{c_i : \forall c_i \in C_N\}, D_N, CF_N, \exists CF_B \leftrightarrow C_N \in \varnothing >, \quad (2.1)$$

де N – ідентифікатором вузла;

$R_N$ – ідентифікатором правила, за яким розраховується вузол N;

$P_N$ – батьківським вузлом N;

$C_N$ – множиною дочірніх вузлів N;

$D_N$ – множиною залежностей, які необхідно мати для розрахунку вузла N за правилом $R_N$;

$CF_N$ – поточним розрахованим ФВ вузла N;

$CF_B$ – поточним розрахованим ФВ шляху (гілки дерева), до якої входить вузол N за умови, що вузол N є листям дерева.

Формальне визначення множини залежностей $D_N$ вузла у (2.1) має вигляд:

$$\{< f_j, \{T_{fk}\} >, \forall f_j \in F_{RN}\}, \quad (2.2)$$

де $\{T_{fk}\}$ – множиною кортежів зі структурою (2.3);

$F_{RN}$ – множина необхідних фактів, що наявні у правилі $R_N$ вузла N.

Формальне визначення кортежу $T_f$ має вигляд:

$$< r, CF >, r \in KB_f, \quad (2.3)$$

де $KB_f$ – множина правил для визначення факту f;

CF – поточним ФВ правила r.

ГЗ (GD) є структурою, яка надає можливість відслідковувати вже знайдені правила, які необхідні для розрахунку кінцевої цілі. Що прибирає необхідність повторного пошуку вже знайденого правила.

Формальне визначення структури ГЗ має вигляд:

$$\{ <f_j, \{ r_k, \forall r_k \in KB_f \}>, \forall f_j \in F \}, \quad (2.4)$$

де F – множиною всіх фактів, що використовуються у системі;

$KB_f$ – множина правил для визначення факту f.

Вхідними даними алгоритму є: БЗ, що містить у собі правила формату «ЯКЩО-ТО»; БД, що містить відомі факти на момент початку використання алгоритму; цільовий факт (goal fact) – факт, для якого необхідно знайти шляхи його розрахунку.

Ідея алгоритму наступна: з черги отримується факт для пошуку відповідних правил для розрахунку. У випадку, коли отриманий факт – цільовий, створюється нове дерево шляхів розрахунку цільового факту, використовуючи знайдені правила як корені дерев. Інакше відбувається пошук вузлів дерев, які потребують отриманий факт для розрахунку, та їх доповнення знайденими правилами. Після доповнення вузлів відбувається уточнення ФВ відповідних шляхів.

Шляхи розрахунку цільового факту отримуються із використанням листя дерев, за умови, що всі їх залежності були знайдені. Якщо хоча б до одної залежності не було знайдено відповідного правила, то шлях вважається недійсним й не бере участь у розрахунку КФВ.

Таким чином, результатом роботи алгоритму є:

- множина шляхів для розрахунку цільового факту з розрахованим ФВ, де кожен шлях є впорядкованим списком правил, які необхідно послідовно розрахувати для отримання цільового факту, проте на момент завершення роботи алгоритму кожен шлях матиме розрахований ФВ цільового факту;
- КФВ, який розраховано на основі ФВ кожного дійсного шляху.

Загальний вигляд алгоритм у форматі блок-схеми наведено на Рис. 2.1.

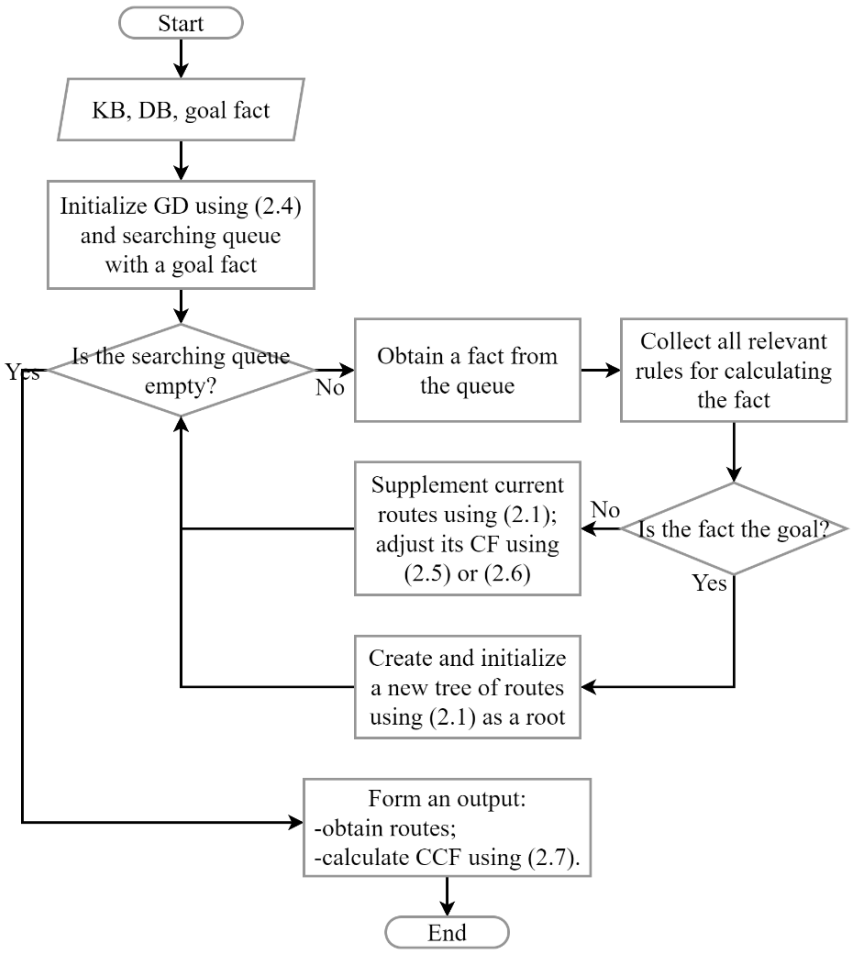



Рисунок 2.1 – Блок-схема алгоритму

В залежності від структури правил у БЗ розрахунок ФВ відбувається за (2.5) чи (2.6). В обох випадках КФВ розраховується за (2.7).

$$cf(H, E_1 \cap E_2 \cap \ldots \cap E_n) = \min[cf(E_1), cf(E_2), \ldots, cf(E_n)] * cf; \quad (2.5)$$

$$cf(H, E_1 \cup E_2 \cup \ldots \cup E_n) = \max[cf(E_1), cf(E_2), \ldots, cf(E_n)] * cf, \quad (2.6)$$

де H – гіпотезою, або ж фактом, що розраховується;

$E_n$ – множина необхідних для розрахунку фактів;

cf – базовим ФВ гіпотези H;

$cf(E_i)$ – ФВ факту $E_i$.

$$cf(cf_1, cf_2) = \begin{cases} cf_1 + cf_2 * (1 - cf_1) & cf_1 > 0 \text{ and } cf_2 > 0 \\ \dfrac{cf_1 + cf_2}{1 - \min[|cf_1|, |cf_2|]} & cf_1 < 0 \text{ or } cf_2 < 0 \\ cf_1 + cf_2 * (1 + cf_1) & cf_1 < 0 \text{ and } cf_2 < 0 \end{cases} \quad (2.7)$$

Деталізований алгоритм комбінованого зворотного виводу наведений у лістингах 2.1-2.4. Перевірку роботи алгоритму продемонстровано у лістингу Б.1.

У лістингу 2.1 подано опис вхідних даних для запуску алгоритму (БЗ, БД, цільовий факт), очікуваного результату його виконання та першого етапу роботи, що полягає в ініціалізації необхідних структур для його коректного функціонування.

Лістинг 2.1 – Етап ініціалізації алгоритму

```
   Input knowledge base (KB), database (DB), a goal.
   Output set of routes for the goal calculation where every route
has precalculated goal certainty factor; combined certainty factor
of the routes.
start BCA
Step 1. Primary initialization
   initialize GD using (2.4)
   for each fact from DB do
```

```
      put <fact,{0}> into GD
   end for each
   put <goal, {}> into GD
   put goal into queue
```

Після завершення етапу ініціалізації виконується пошук правил (лістинг 2.2), які дозволяють вивести новий факт, тобто поточну під ціль. На цьому етапі можна зробити невеликі зміни до алгоритму для зміни стратегії пошуку правил. За замовчуванням використовується стратегія «всі підходящі», яка передбачає пошук усіх правил, що можуть допомогти досягти поточної під цілі. Однак, якщо потрібно застосувати стратегію «перший підходящий», слід перервати цикл у зазначеному місці лістингу 2.2.

Лістинг 2.2 – Початок основного циклу алгоритму та пошук правил

```
Step 2. Main algorithm loop
   while queue is not empty do
      pop fact from queue
      initialize rule_container
Step 2.1. Searching for necessary rules
      while r ≤ R, where r = 1, 2, …, R, R is a number of rules in
      KB do
         if Rule_r from KB gives fact then
            append <fact, r> into GD
            for each required_fact from Rule_r do
               if required_fact is not in GD then
                  put required_fact into queue
                  put <required_fact, {}> into GD
               end if
            end for each
            append r into rule_container
            //If a 'first-match' strategy is required it's the
            point of breaking the while cycle
            //otherwise 'all-match' strategy is implemented
         end if
      end while
      if fact is goal then
```

У лістингу 2.3 описано основну частину алгоритму, яка передбачає створення нових дерев розрахунків у випадку, коли поточна під ціль відповідає цільовому факту алгоритму, що була передана як вхідний параметр.

Якщо ж поточна під ціль є проміжним фактом, здійснюється доповнення та оновлення існуючих дерев. На цьому етапі застосовуються допоміжні алгоритми-функції, що описано у лістингах 2.4-2.6.

Лістинг 2.3 – Створення дерева рішень з розрахунком ФВ

```
Step 2.2. Creating and initialization of a new tree of routes
         for each r in rule_container do
            create new_node using Rule_r and (2.1)
            create new_tree using new_node as a root
            append new_tree to tree_container
         end for each
      else then
Step 2.3. A supplement current routes and an adjustment of its
certainty factors
        for each tree in tree_container do
           for each node in tree do
              if fact is in dependencies of node then
                 find leaves of the node using GetLeaves(node)
                 (List. 2.5) algorithm
                 for each leaf in leaves do
                    for each r in rule_container do
                       create new_node using Rule_r and (2.1): {R_N
                          is r, P_N is leaf, C_N is{}, D_N
                          initialization of the dependencies using
                          Rule_r, GD and (2.2), CF_N is the base CF
                          of Rule_r, CF_B is inherited from leaf}
                       set CF_B of leaf to null
                       if Rule_r contains known required_facts from
                       GD, but not from DB then
                          set new_node to a result of
                          RestorePath(Rule_r) (List. 2.6) algorithm
                       end if
                       set update_from to a result of
                       GetLeaves(node) algorithm
                       for each start_node from update_from do
                          set CF_B of start_node to result of
                             UpdateAncestors(start_node) (List.
                             2.7) algorithm
                       end for each
                    end for each
                 end for each
                 update leaves of tree
              end if
           end for each
        end for each
      end if-else
   end while
```

Лістинг 2.4 описує завершальний етап алгоритму, на якому відбувається формування результату його роботи. Ці результати включають КФВ та множину дійсних шляхів досягнення цільового факту алгоритму.

Лістинг 2.4 – Завершальний етап алгоритму

```
Step 3. Forming of an output
   obtain all routes using leaves of tree
   calculate combined_cf using (2.7)
end BCA
```

## 2.2 Опис допоміжних алгоритмів-функцій

Алгоритм модифікованого зворотного виводу також використовує низку допоміжних алгоритмів-функцій. До них входять наступні: GetLeaves, RestorePath, UpdateAncestors.

Алгоритм GetLeaves наведений у лістингу 2.5. Цей рекурсивний алгоритм, використовуючи початковий вузол, повертає множину листків піддерева, де коренем є початковий вузол. Він використовується у випадках, коли необхідно до поточного вузла додати один або декілька нових одного рівня (було знайдено декілька правил для розрахунку факту), проте поточний вузол може мати дочірні вузли. Тому, щоб зберегти коректні шляхи розрахунку цільового факту, необхідно додавати їх до листків піддерева.

Ідея алгоритму-функції наступна: рекурсивно переходити до дочірніх вузлів, доки $C_N$ поточного вузла не дорівнюватиме порожній множині – це кінцева точка рекурсії.

Результатом є множина листків піддерева.

Лістинг 2.5 – Алгоритм-функція GetLeaves

```
Input a start node from which leaves are obtained
Output a set of nodes that represent leaves of start node
```

```
start GLA
   set leaves to {}
   if children of start_node are empty then
      append start_node to leaves
      return leaves
   else
      for each child from children of start_node do
         append the result of GetLeaves(start_node) (List. 2.5) to
         leaves
      end for each
      return leaves
   end if-else
end GLA
```

У момент знаходження правила для розрахунку поточного факту з черги, може трапитися ситуація, коли серед залежностей є ті правила, що вже було знайдено, тому, щоб уникнути повторного знаходження вже знайдених правил, було розроблено алгоритм-функцію RestorePath (лістинг 2.6). Цей рекурсивний алгоритм на вході отримує правило, з якого буде відбуватися відновлення шляхів. Відновлення відбуватиметься до тих пір, доки не відновиться правило, у якому всі залежності – дані з БД, або відповідні правила для необхідних фактів ще не булі знайдені.

Ідея полягає у наступному: відбувається співставлення залежностей правила з ГЗ. У разі, коли значення факту у ГЗ є множина {0}, відбувається отримання його значення ФВ з БД. Інакше, коли значення множини відрізняється та не дорівнює порожній множині, відбувається відновлення кожного вузла з множини.

Результатом алгоритму-функції є оновлений вузол, з якого почалося відновлення шляху.

Лістинг 2.6 – Алгоритм-функція RestorePath

```
   Input R_N of a node from which path is restored
   Output an updated node from which path is restored
start RPA
Step 1. Primary initialization
   set child_layer to null
```

```
   initialize restored_node using R_N and (2.1): {R_N is R_N from
      input, P_N is null, C_N is {}, D_N initialization of the
      dependencies using Rule_r and (2.2) with null, CF_N is the base
      CF of Rule_r, CF_B is null}
   for each required_fact from R_N do
Step 2. Determine the source of a fact
      if a set of rules of required_fact in GD is equal to {0}
      then
         set required_fact in D_N of restored_node to <0, CF_N from
         DB>
      else if a set of rules of required_fact in GD isn't equal to
      null then
         if child_layer is null then
            set child_layer to required_fact
         end if
Step 3. Restore nodes of a fact layer
         set layer_nodes to {}
         for each r in a set of rules of required_fact in GD do
            append result of RestorePath(r) to layer_nodes
            append <r, null> to a set of rules of required_fact in
               D_N of restored_node
         end for each
Step 4. Restore relations between nodes of the fact layer and the
restored node
         if child_layer is required_fact then
            set P_N of each from layer_nodes to restored_node
            set C_N of restored_node to layer_nodes
         else
            find leaves of restored_node using
               GetLeaves(restored_node) (List. 2.5) algorithm
            for each leaf from leaves do
               set P_N of each from layer_nodes to leaf
               set C_N of leaf to layer_nodes
            end for each
         end if-else
      end if-else
   end for each
   return restored_node
end RPA
```

Після додавання нового вузла або відновлення шляху відбувається уточнення ФВ починаючи з листів піддерева. Алгоритм-функція оновлення наведена у лістингу 2.7.

Ідея полягає у наступному: через те, що кожен факт може мати декілька правил для розрахунку, використовується множина кортежів, що має структуру (2.3) для ідентифікації відповідного правила у залежностях вузла для оновлення. Починаючи з вузла-листя, відбувається поступовий перехід до

батьківських вузлів оновлюючи їх залежності та доповнюючи множину кортежів.

Лістинг 2.7 – Алгоритм-функція UpdateAncestors

```
   Input a start node from which ancestors are updated
   Output certainty factor of a branch
start UAA
   set arguments with {} that have (2.3) structure
Step 1. Updating start node's certainty factor
   recalculate start_node's CF_N
   append <R_N, CF_N> of start_node to arguments
Step 2. Update ancestor nodes
   set current_node to start_node
   while P_N of current_node exists do
      set current_node to P_N of current_node
      if dependencies of current_node have any from arguments do
         update dependencies using R_N of arguments' to identify
         corresponding ones
         recalculate current_node's CF_N
      end if
      append <R_N, CF_N> of current_node to arguments
   end while
   return CF_N of last added argument
end UAA
```

Результатом є ФВ цільового факту використовуючи відповідний шлях розрахунку.

# 3 ТЕСТУВАННЯ РОБОТИ АЛГОРИТМУ

Нехай місією для тестування роботи алгоритму буде повторюване перевезення певного вантажу роботом з позиції 2 на позицію 5 (рис. 3.1). Для досягнення мети робот має подолати певні етапи, тобто досягти виконання локальних цілей. В даному випадку, локальні цілі АІБС можуть бути представлені певними етапами плану реалізації місії, тобто послідовністю станів системи «Оточення-АІБС», які система має виконати завдяки своїм діям.

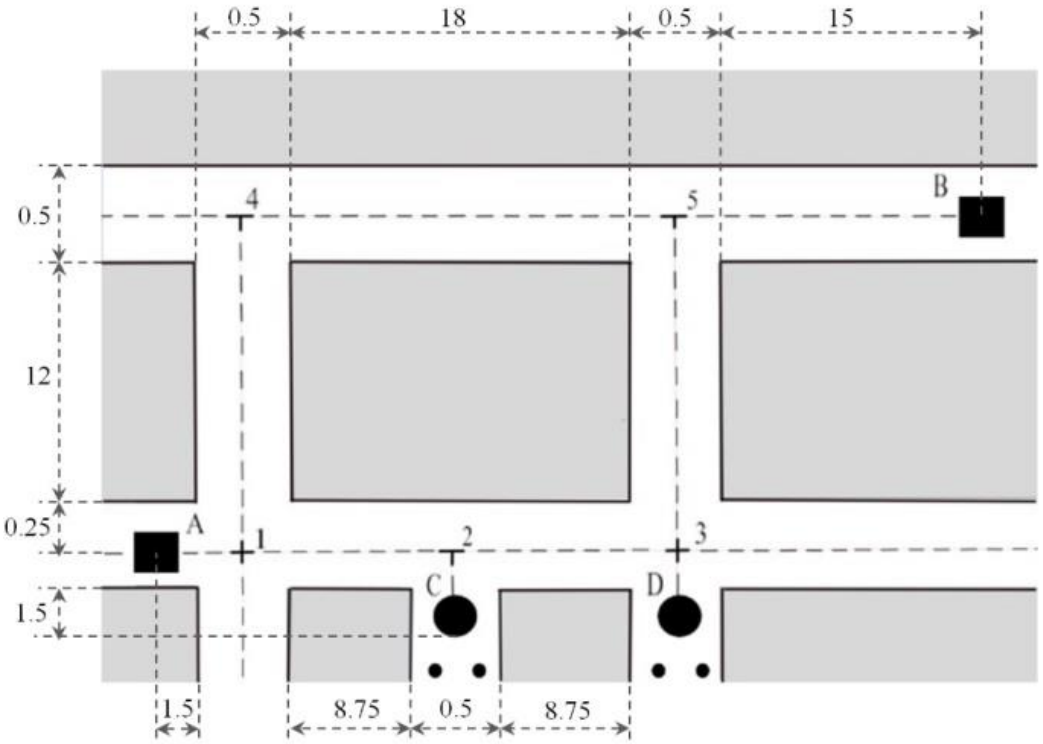



Рисунок 3.1 – Модель оточення АІБС

У прикладі роботи алгоритму на рис. Б.1 на вхід алгоритму подається БЗ, яка містить набір правил, що наведено у (Б.1). У якості вмісту БД було обрано початкове положення робота $\{^{*}2^{\text{Move}}\}$ з ФВ 0.9, а у якості цілі – необхідне кінцеве положення $^{*}5^{\text{Move}}$.

Крок 1) на рис. Б.1 виконується відповідно **Step 1** базового алгоритму у лістингу 2.1, де відбувається занесення цілі до черги *queue*, ініціалізація *tree_container* та GD використовуючи БД та ціль.

Починаючи з кроку 2) відбувається основний цикл алгоритму **Step 2**, де з черги *queue* вилучається факт з його занесенням до *fact* та ініціалізується *rule_container*. Наступним є **Step 2.1**, результатом виконання якого є: заповнений *rule_container* з правилами, які дають змогу отримати *fact*; доповнення черги *queue* фактами, правила для яких ще не було знайдено; оновлення GD, використовуючи знайдені правила та невідомі факти. На кроці 2) було: знайдено правила {3, 22, 28} для отримання факту $^{*}5^{\text{Move}}$; додано два невідомі факти до *queue*: $^{*}4^{\text{Move}}$, $^{*}3^{\text{Move}}$ та оновлено GD.

Через те, що 2) – це перша ітерація алгоритму – відбувається виконання **Step 2.2**: створення дерев та використання знайдених правил для створення їх коренів.

Наприклад, коренем першого дерева tree_1 є вузол з: $N = 0$, що є ідентифікатором вузла; $R_N = 3$, що вказує на правило, за яким розраховується вузол; $P_N = \text{null}$, що означає відсутність батьківського вузла; $C_N = \{\}$, що означає відсутність дочірніх вузлів; $D_N = \{^{*}4^{\text{Move}}, \text{null}>\}$ вказує, що $^{*}4^{\text{Move}}$ є необхідним для розрахунку правила фактом, для якого ще не біло знайдено відповідних правил; $CF_N = 0.9$ є ФВ вузла, що дорівнює базовому ФВ правила (через те, що не всі залежності було знайдено); $CF_B = 0.9$ – ФВ поточного шляху розрахунку. Таким чином відбувається створення та ініціалізація дерев.

Кроки з 3) по 6), включно, мають ідентичну логіку виконання за винятком, що замість **Step 2.2** виконується **Step 2.3** (всі наступні шукані факти не є ціллю). Таким чином, наприклад у кроці 3), відбувається пошук такого вузла дерева, у якому факт $^{*}4^{\text{Move}}$ входить до його залежностей $D_N$, після чого отримуються листя піддерева (за допомогою алгоритму GetLeaves – лістинг 2.5) для коректного додавання нових вузлів (в даному випадку такий вузол, корінь дерева, і є листям) – створюються нові вузли (у кроці 3) тільки один

вузол створено), оновлюються зв’язки між батьківським та дочірніми вузлами. Відбувається оновлення $D_N$ батьківського вузла.

На кроці 4) можна побачити процес уточнення ФВ. Через те, що правила 15, 21 (вузли 4, 5, 7, 8) мають серед залежностей факт з БД, то є можливість одразу їх оновити. Таким чином, пожинаючи з нового вузла дерева та використовуючи алгоритм оновлення предків UpdateAncestors (лістинг 2.7), відбувається уточнення ФВ.

На кроці 6) можна побачити процес відновлення шляху за допомогою алгоритму RestorePath (лістинг 2.6) коли, наприклад, після створення вузла 14 за правилом 18 можна спостерігати ситуацію, що серед залежностей є факт ${}^{*}1^{\mathrm{Move}}$, правила розрахунку якого вже було знайдено – відбувається відновлення.

Між кроками 6) та 7) також відбувається пошук правил для факту ${}^{*}\mathrm{A}^{\mathrm{Load}}$, однак таких правил не існує, тому змін не відбувається.

Крок 7) є завершальним, на якому виконується **Step 3**, де формується множина шляхів використовуючи листя дерев (початок шляху), серед яких обираються дійсні шляхи базуючись на тому, що початковий вузол (листя дерева) повинен мати знайденими всі залежності. На основі дійсних шляхів розраховується КФВ (2.7) (кожен початковий вузол шляху має ФВ цільового факту при використанні цього шляху).

Отже, результатом роботи алгоритму є множина шляхів *Valid_routes*, яка містить 6 рекомендованих шляхів із розрахованими ФВ { {15>22}(0.729), {21>22}(0.729), … } та КФВ *Combined_certainty_factor*, що дорівнює 0.9992. Кожен шлях є строгою для виконання послідовністю правил як, наприклад, шлях {15>22}, де 15 – це номер правила у (Б.1), з якого починається розрахунок (листя дерева), а 22 – номер останнього правила у (Б.1) (корінь дерева), розрахувавши яке буде отримано ціль ${}^{*}5^{Move}$ з ФВ 0.729.

Таким чином, при виконанні цієї послідовності правил, робот з початкового положення ${}^{*}2^{Move}$ спочатку переміститься до положення ${}^{*}3^{Move}$ (правило 15), а потім – до цілі ${}^{*}5^{Move}$ (правило 22) з ФВ 0.729.

# 4 ОПИС МОДЕЛІ ПРОТОТИПУ СИСТЕМИ ДЛЯ ТЕСТУВАННЯ

## 4.1 Опис компонентів апаратної частини моделі прототипу системи

У якості фізичної моделі системи для створення прототипу АІБС було обрано робот на колісній базі.

До складу робота входять:

- шасі на базі колес Mecanum (рисунок 4.1);
- мікроконтролер Arduino Mega 2560 (рисунок 4.2);
- WiFi модуль ESP8266-01 (рисунок 4.3);
- плата роширення Arduino Motor Shield із контролером керування електродвигунами (рисунок 4.4);
- датчики відображення KY-033 (рисунок 4.5);
- ультразвукові датчики відстані HC-SR04 (рисунок 4.6).

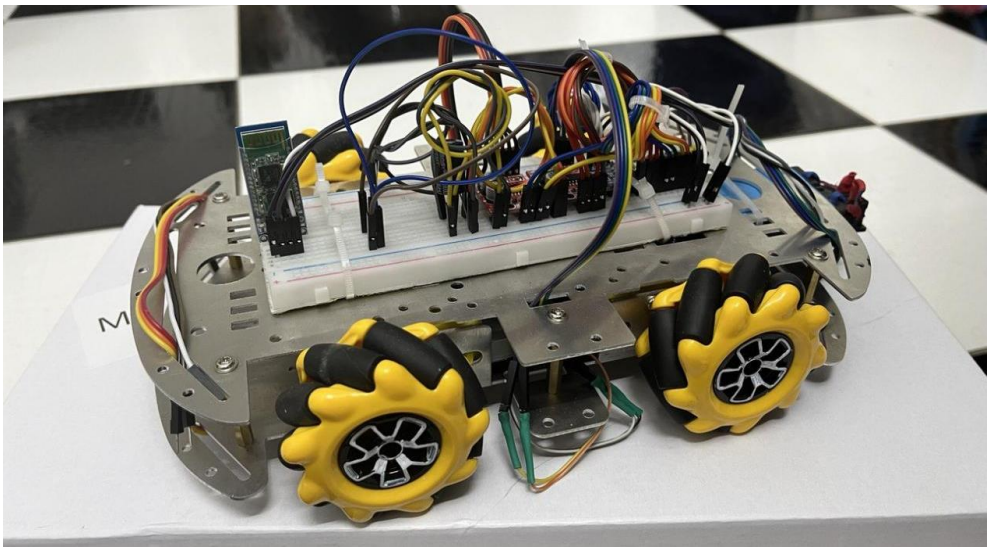


Рисунок 4.1 – Чотирьох колісне шасі

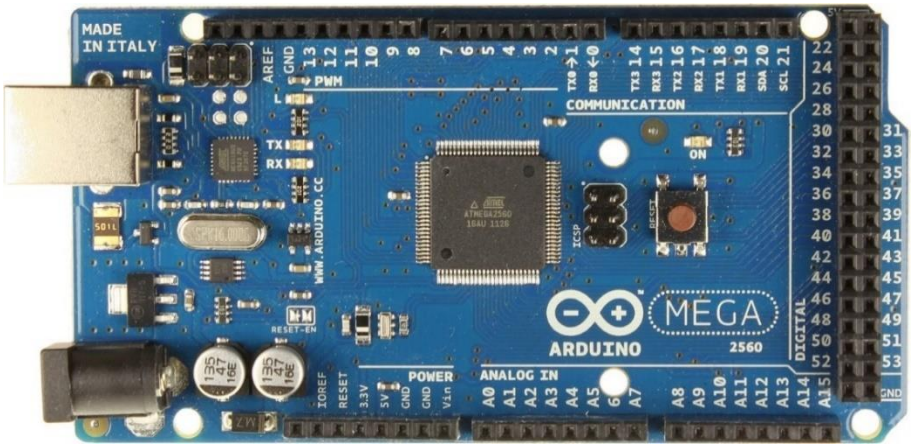



Рисунок 4.2 – Мікроконтролер Arduino Mega 2560

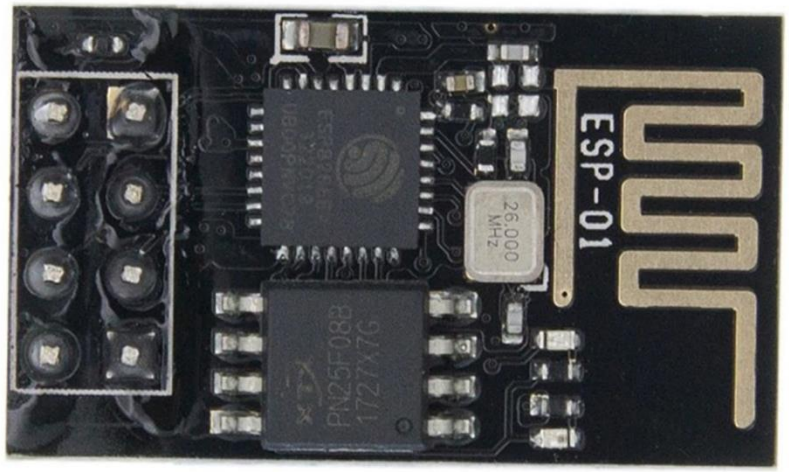



Рисунок 4.3 – WiFi модуль ESP8266-01

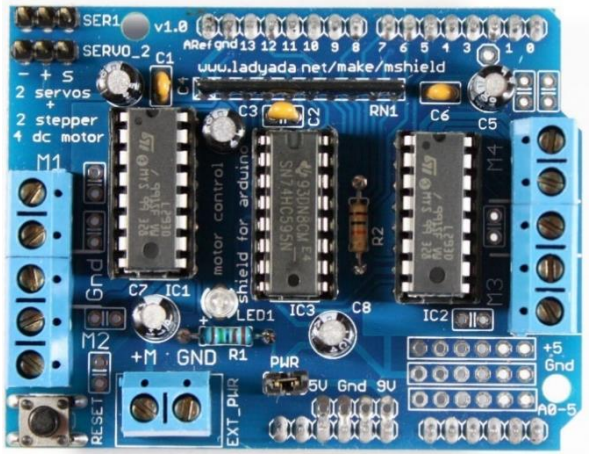



Рисунок 4.4 – Плата розширення Arduino Motor Shield

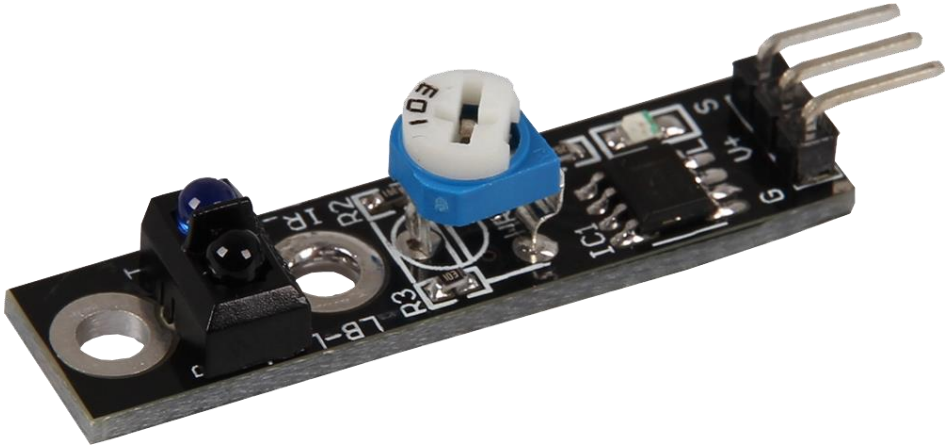


Рисунок 4.5 – Датчик відображення KY-033

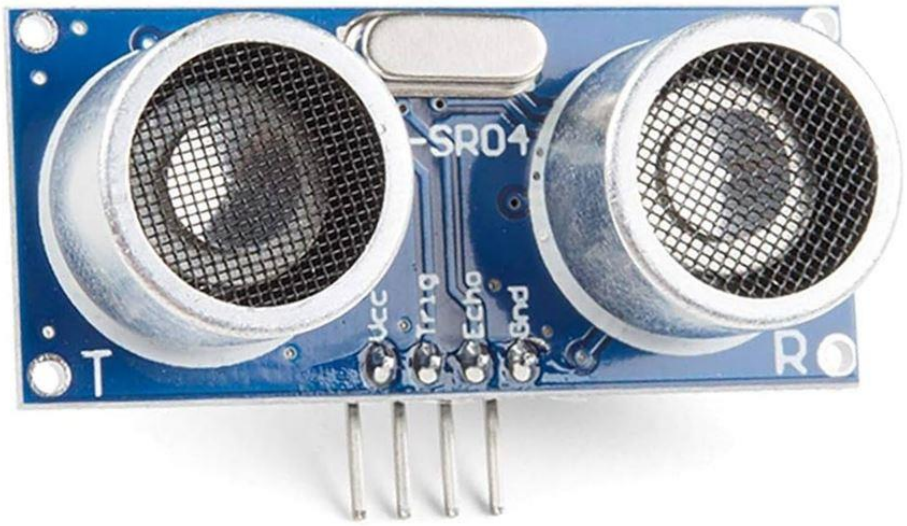



Рисунок 4.6 – Ультразвуковий датчик HC-SR04

### 4.1.1 Опис мікроконтролера Arduino Mega 2560

Arduino Mega 2560 представляє собою високопродуктивну плату мікроконтролера на базі ATmega2560, яка призначена для реалізації складних проектів, що потребують значної кількості вхідних та вихідних контактів. Плата підтримує широкий спектр додаткових модулів та плат розширення, за допомогою яких є можливість збільшити функціональні можливості. Основні характеристики наведені у таблиці 4.1.

Таблиця 4.1 – Основні характеристики Arduino Mega 2560

| Характеристика | Значення |
|---|---|
| Мікроконтролер | ATmega256 |
| Кількість цифрових I/O | 54 (з них 15 PWM) |
| Кількість аналогових I/O | 16 |
| Flash пам'ять | 256 KB |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Робоча напруга (живлення) | 5 V (7-12 V) |
| Тактова частота | 16 MHz |
| Максимальний струм одного I/O | 40 mA |
| Порти UART | 4 |

### 4.1.2 Опис WiFi модулю ESP8266-01

ESP8266-01 – є базовим Wi-Fi модулем, що дозволяє додавати бездротове підключення до проектів на базі мікроконтролерів. Модуль може функціонувати як самостійний мікроконтролер або у поєднанні з іншими контролерами, такими як Arduino. Основні характеристики наведені у таблиці 4.2.

Таблиця 4.2 – Основні характеристики ESP8266-01

| Характеристика | Значення |
|---|---|
| Мікроконтролер | ESP8266 |
| Робоча напруга | 3.3V |
| Flash пам'ять | 1 MB |
| Порти GPIO | 2 |
| Робоча частота | 80 MHz |
| Стандарти Wi-Fi | 802.11 b/g/n |
| Протоколи безпеки | WPA/WPA2 |
| Інтерфейси | UART, SPI |
| Антена | Вбудована або зовнішня |

### 4.1.3 Опис плати розширення Arduino Motor Shield

Arduino Motor Shield являє собою додаткову плату для Arduino, яка дозволяє керувати різними типами двигунів. Плата розширення побудована на основі драйвера L298P і дозволяє підключати та керувати чотирьома двигунами постійного струму або одним кроковим двигуном. Основні характеристики наведені у таблиці 4.3.

Таблиця 4.3 – Основні характеристики Arduino Motor Shield

| Характеристика | Значення |
|---|---|
| Мікросхема драйвера | L298P |
| Кількість каналів | 4 |
| Максимальна напруга | 12V |
| Максимальний струм | 2A |
| PWM керування | Так |

### 4.1.4 Сенсор відображення KY-033

KY-033 представляє собою інфрачервоний сенсор для виявлення лінії. Він включає інфрачервоний випромінювач та приймач, що дозволяє йому визначати колір поверхні (чорна або біла лінія). Основні характеристики наведені у таблиці 4.4.

Таблиця 4.4 – Основні характеристики KY-033

| Характеристика | Значення |
|---|---|
| Датчик | ІЧ випромінювач та приймач |
| Робоча напруга | 3.3V - 5V |
| Діапазон виявлення | 0.1 - 2 см |
| Цифровий вихід | Так (налаштовується за допомогою потенціометра) |
| Антибліковий фільтр | Так |

### 4.1.5 Сенсор відстані HC-SR04

HC-SR04 є ультразвуковим сенсором відстані, що дозволяє з високою точністю визначати відстань до об'єктів. Основні характеристики наведені у таблиці 4.5.

Таблиця 4.5 – Основні характеристики датчика KY-033

| Характеристика | Значення |
|---|---|
| Робочий струм | 15mA |
| Робоча напруга | 5V |
| Діапазон вимірювання | 2 см - 400 см |
| Точність | 3 мм |
| Кут виявлення | 15 |

## 4.2 Опис архітектури апаратної частини моделі прототипу системи

На рис. 4.7 продемонстровано приклад з’єднання 10 сенсорів відображення, що дозволить роботу автономно слідкувати за обраним маршрутом та оминати невеликі перешкоди. Для цього їх було підключено до цифрових портів плати Arduino Mega з номерами 23-32. Вони будуть сигналізувати про наявність розмітки під ними зміненням рівня сигналу з низького (логічний 0) на високий (логічна 1).

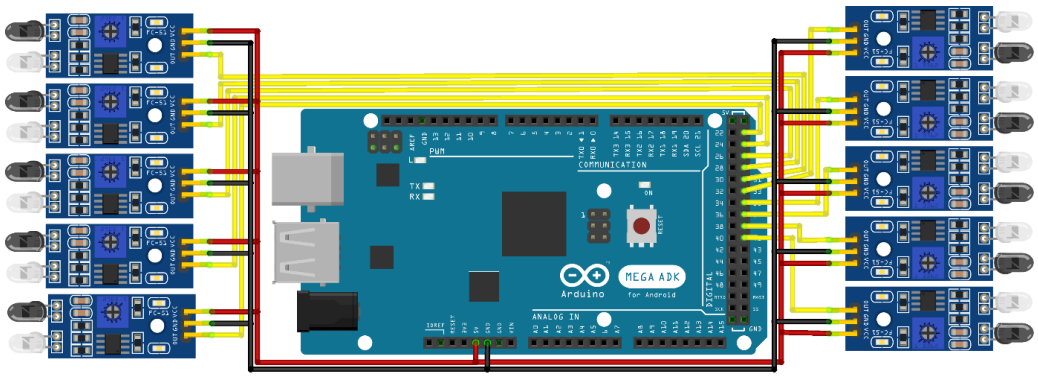



Рисунок 4.7 – Схема з’єднання компонентів моделі прототипу: 10 датчиків відображення

На рис. 4.8 продемонстровано приклад з’єднання 4 ультразвукових сенсори відстані, що дозволять роботу автономно слідкувати, краще орієнтувати у своєму оточенні та оминати перешкоди. Для цього кожен вихід сенсорів Trig було з’єднано в єдину шину для ініціалізації початку розрахунку відстані (порт номер 22), а для отримання даних з сенсорів кожен вихід Echo було з’єднано з окремим цифровим портом плати, що підтримує ШІМ.

Цей датчик визначає відстань до об'єкту, вимірюючи час між надсиланням та отриманням звукової хвилі від об'єкту. Для забезпечення концентрування напрямку звукової хвилі частоту звукової хвилі було покладено в межі ультразвуку, це є можливим через менше розсіювання високочастотного звуку у середовищі.

Для забезпечення зв’язку роботу із сервером використовується WiFi модуль, який з’єднується з виходами одного UART на платі. Вони мають номери 18 TX1 та 19 RX1, які під’єднуються до відповідних виходів на платі модуля, що продемонстровано на рисунку 4.9.

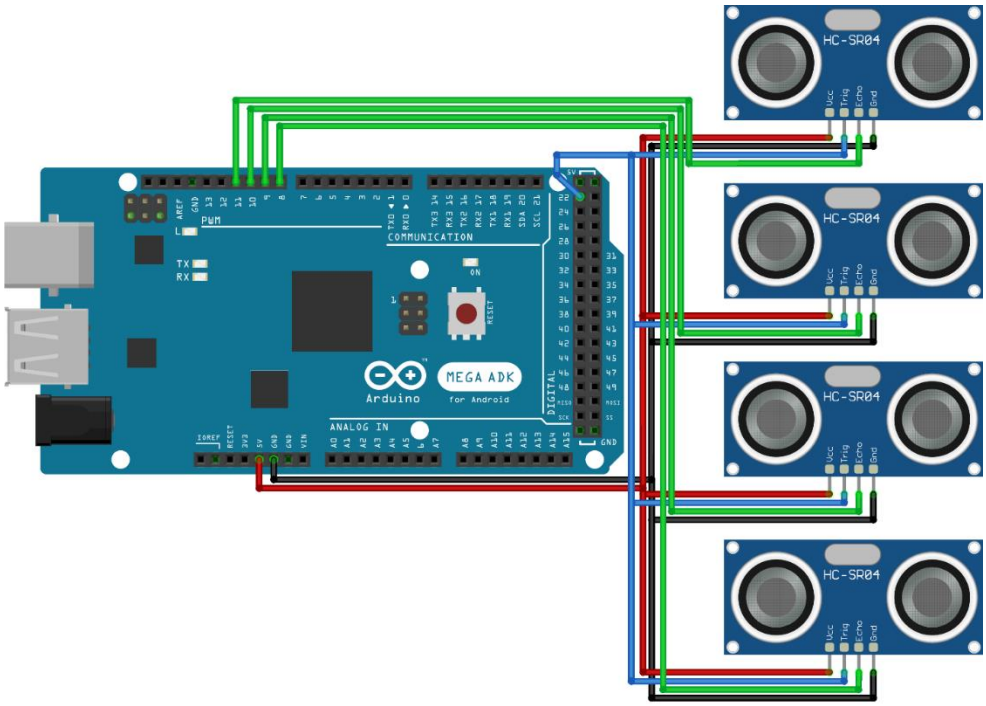



Рисунок 4.8 – Схема з’єднання компонентів моделі прототипу: 4 ультразвукових датчики відстані

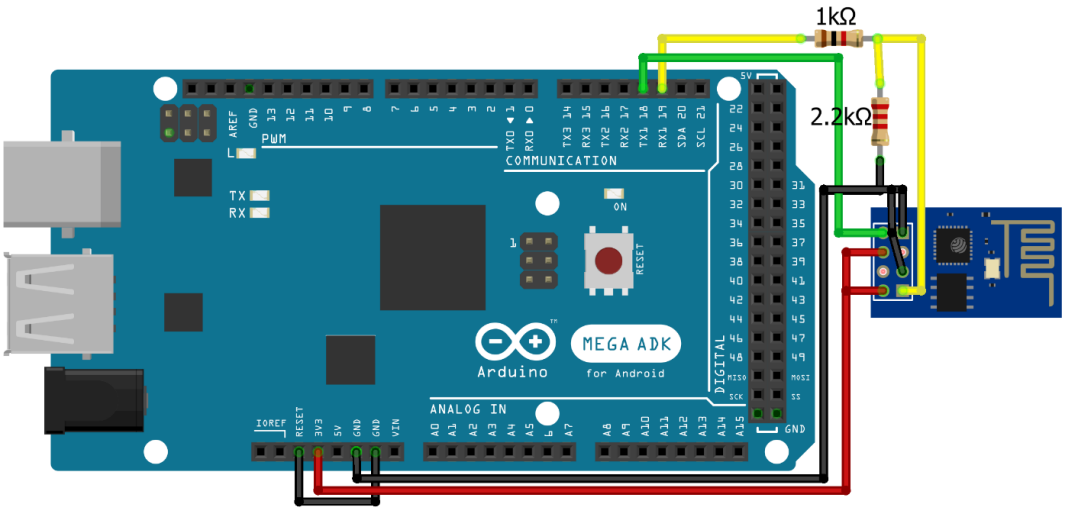



Рисунок 4.9 – Схема з’єднання компонентів PM: WiFi модуль ESP8266-01

4.3 Опис архітектури та компонентів програмної частини прототипу системи

На даному етапі передбачається, що основний обсяг розрахунків буде покладено на віддалений сервер, який буде взаємодіяти із роботом через мережі WiFi, схему чого продемонстровано на рисунку 4.10.

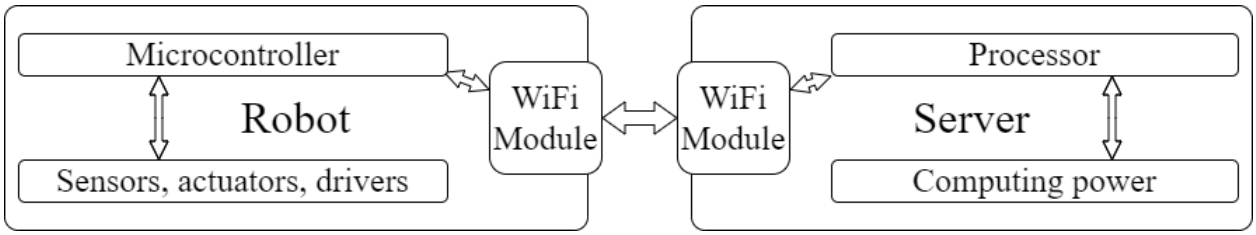



Рисунок 4.10 – Модель взаємодії робота з сервером

У лістингу 4.1 приведено абстрактний приклад впровадження алгоритму неперервного планування (рисунок 4.11), який базується на взаємодії робота із віддаленим сервером.

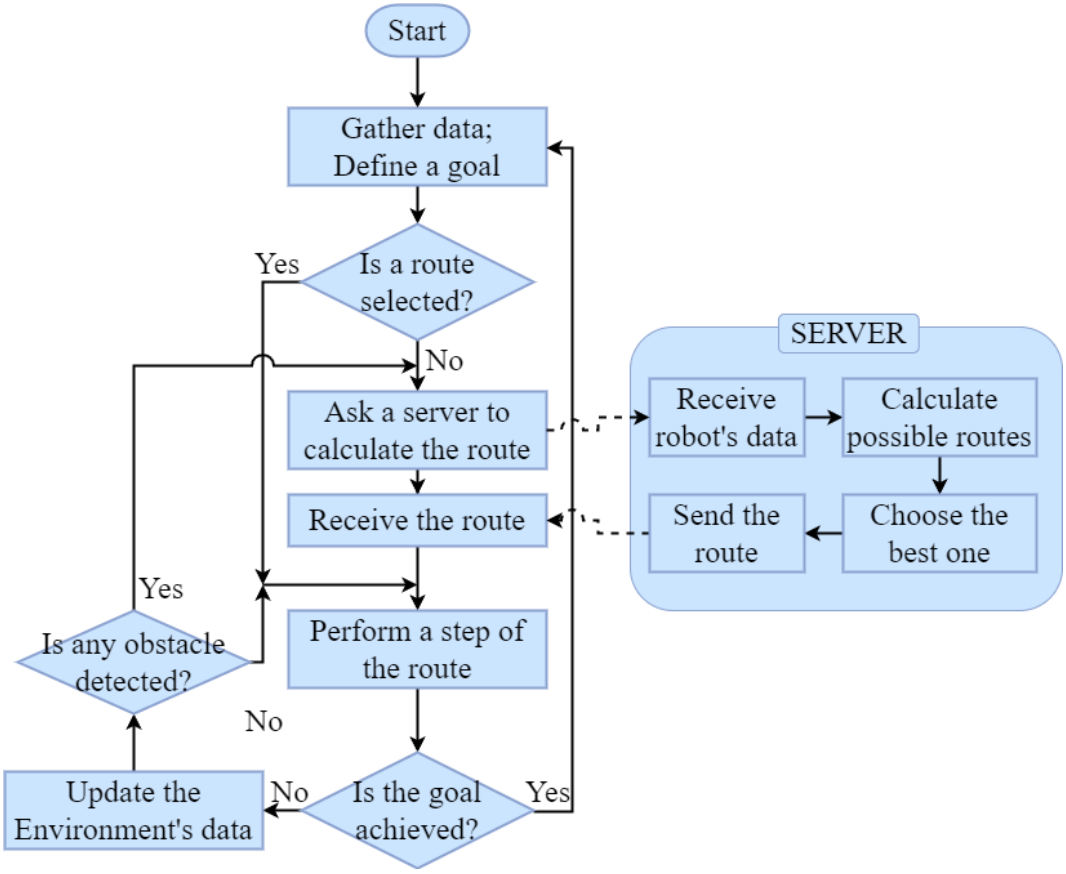



Рисунок 4.11 – Алгоритм неперервного планування

Лістинг 4.1 – Абстрактний шаблон програми управлінням АІБС

```
#define Sensor_Ref_one 22 //Оголошення сенсорів по номеру (23-32)
…
#define Sensor_Ref_ten 31
#define WiFiTransceive 18
#define WiFiReceive 19
void loop() {
   //Збір інформації з сенсорів
   int value_one = digitalRead(Sensor_Reg_one);
   …
   int value_ten = digitalRead(Sensor_Ref_ten);
   //Місце впровадження алгоритму неперервного планування
   char* receiveData = WiFiReceive.ReceiveData();
   robot.performStage(receiveData);
   delay(50);}
```

Сервер розраховує шлях, базуючись на алгоритмі представленому у розділі 2, та надсилає його роботу через мережу WiFi.

# ВИСНОВКИ

Результатом кваліфікаційної роботи є система комбінованого нечіткого виведення для подальшого впровадження у вбудовані системи.

Для розробки системи було виконано зроблено наступне:

- було розроблено алгоритм комбінованого зворотного виводу;
- було описано апаратну та програмну модель прототипу системи.

Розроблений алгоритм передбачає наступні пункти:

- алгоритм приймає у якості вхідних даних: БЗ у форматі правил «ЯКЩО-ТО», БД із відомими фактами, кінцевий факт як ціль пошуку;
- алгоритм відстежує знайдені проміжні факти для зменшення кількості обчислень;
- алгоритм уточнює ФВ шляхів у процесі їх доповнення знайденими необхідними фактами;
- алгоритм має можливість відновлення шляху, якщо вже були знайдені відповідні правила;
- результатом роботи алгоритму є множина шляхів для розрахунку цільового факту з розрахованими ФВ кожного шляху та КФВ.

Опис апаратної та програмної моделі прототипу системи описує наступне:

- компоненти апаратної частини моделі прототипу системи;
- архітектуру апаратної частини моделі прототипу системи;
- архітектуру та компоненти програмної частини прототипу системи;
- алгоритм неперервного планування;
- шаблон програмного управління АІБС.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kargin A., Hiievskyi D., Oliinyk D. Безперервне планування і ситуаційне управління як завдання штучного інтелекту що відчуває. Системи управління, навігації та зв'язку. Збірник наукових праць. 2024. Т. 2, № 76. С. 106–110. URL: https://doi.org/10.26906/sunz.2024.2.106

2. Robotics: State of the art and future challenges / G. Bekey et al. London: Imperial College Press, 2008. 152 p. URL: https://doi.org/10.1142/p542

3. Industry 4.0 vision for energy and materials / ed. by M. Sharifzadeh. Wiley, 2022. URL: https://doi.org/10.1002/9781119695868

4. Tan Y., Duan F., Li W. Intelligent robot: implementation and applications. Springer, 2023. 302 p. URL: https://doi.org/10.1007/978-981-19-8253-8

5. International Organization for Standardization. (2021). Robotics – Vocabulary (ISO Standard No. ISO 8373:2021). https://www.iso.org/standard/75539.html

6. Sotnik S., Lyashenko V. Agricultural robotic platforms. International Journal of Engineering and Information Systems. 2022. Vol. 6, no. 4. P. 14–21.

7. Development status and trend of agricultural robot technology / Y. Jin et al. International Journal of Agricultural and Biological Engineering. 2021. Vol. 14, no. 3. P. 1–19. URL: https://doi.org/10.25165/j.ijabe.20211404.6821

8. IEEE SPECTRUM, Guizzo E. Types of robots. Robots: your guide to the world of robotics. 2023 May 23. URL: https://robotsguide.com/learn/types-of-robots

9. A review of the agricultural robot as a viable device for productive mechanized farming / T. J. Erinle et al. 2021. URL: https://doi.org/10.31219/osf.io/wgc54

10. An overview of perception methods for horticultural robots: from pollination to harvest / H. S. Ahn et al. arXiv(Cornell University). 2018. URL: https://doi.org/10.48550/arXiv.1807.03124

11. Kargin A., Petrenko T. Spatio-temporal data interpretation based on perceptional model. advances in spatio-temporal segmentation of visual data. Cham, 2019. P. 101–159. URL: https://doi.org/10.1007/978-3-030-35480-0_3

12. Negnevitsky M. Artificial intelligence: a guide to intelligent systems (2nd edition). 2nd ed. Addison Wesley, 2004. 440 p.

13. Al-Ajlan A. The comparison between forward and backward chaining. International Journal of Machine Learning and Computing. 2015. Vol. 5, no. 2. P. 106–113. URL: https://doi.org/10.7763/ijmlc.2015.v5.492

14. Poole D., Mackworth A. Artificial intelligence. Foundations of computational agents. 3rd ed. Cambridge : Cambridge University Press, 2023. 900 p. URL: https://doi.org/10.1017/9781009258227.022

15.

16. Sariff N., Buniyamin N. An overview of autonomous mobile robot path planning algorithms. 2006 4th Student Conference on Research and Development, Shah Alam, Malaysia, 27–28 June 2006. URL: https://doi.org/10.1109/scored.2006.4339335

17. Mohanty P., Parhi D. Controlling the motion of an autonomous mobile robot using various techniques: a review. Journal of Advanced Mechanical Engineering. 2013. URL: https://doi.org/10.7726/jame.2013.1003

18. Kargin A., Panchenko S., Petrenko T. Implementation of cognitive perception functions in fuzzy situational control model. Procedia Computer Science. 2019. Vol. 149. P. 231–238. URL: https://doi.org/10.1016/j.procs.2019.01.128

19. Alatise M. B., Hancke G. P. A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access. 2020. Vol. 8. P. 39830–39846. URL: https://doi.org/10.1109/access.2020.2975643

20. A Taxonomy for mobile robots: types, applications, capabilities, implementations, requirements, and challenges / U. Jahn et al. Robotics. 2020. Vol. 9, no. 4. P. 109. URL: https://doi.org/10.3390/robotics9040109

21. Autonomous robots for services–state of the art, challenges, and research areas / M. Misaros et al. Sensors. 2023. Vol. 23, no. 10. P. 4962. URL:

https://doi.org/10.3390/s23104962

22. Li X. The technology statues and trend of path planning of logistics robot. applied and computational engineering. 2023. Vol. 2, no. 1. P. 563–568. URL: https://doi.org/10.54254/2755-2721/2/20220595

23. Motion control of smart autonomous mobile system based on the perception model / A. Kargin et al. ICTE in Transportation and Logistics 2019. Cham, 2020. P. 145–153. URL: https://doi.org/10.1007/978-3-030-39688-6_20