

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

**КОЛОМІЄЦЬ МАРІЯ ВІКТОРІВНА**

Допускається до захисту:  
Завідувач кафедри  
інформаційних технологій  
к.т.н. Нескородева Т. В.  
« \_\_\_\_ » \_\_\_\_\_ 20\_\_ р.

**СИСТЕМА ВІДДАЛЕНОГО УПРАВЛІННЯ РОБОТОТЕХНІКОЮ НА  
ПЛАТФОРМІ WEBOTS З МОДУЛЬНИМ ТИПОМ ПОВЕДІНКИ ТА SQL  
БАЗОЮ ДАНИХ**

Спеціальність 122 Комп'ютерні науки

**Кваліфікаційна (бакалаврська) робота**

Керівник:

Мартянова Т.А., ст.викладач кафедри  
інформаційних технологій, к.т.н.

Оцінка: \_\_\_\_ / \_\_\_\_ / \_\_\_\_

(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: \_\_\_\_\_

(підпис)

Вінниця 2021

## АНОТАЦІЯ

**Коломієць М.В. "Система віддаленого управління робототехнікою на платформі Webots з модульним типом поведінки та SQL базою даних".** Спеціальність 122 «Комп'ютерні науки», спеціалізація «Розробка програмного забезпечення і комп'ютерна графіка». Донецький національний університет імені Василя Стуса, Вінниця 2021.

У кваліфікаційній (бакалаврській) роботі досліджено специфіку роботи симуляторів, зокрема Webots. Показано переваги і недоліки існуючих інтелектуальних рішень в області існуючих систем керування роботизованими пристроями. Спроековано систему управління робототехнікою на платформі Webots та базу даних під розроблений додаток.

Ключові слова: логічні контролери, контролери для автоматизації, автоматизована рухома платформа (АРП), віддалений блок управління (ВБУ), програмне забезпечення (ПЗ), програмні комплекси для дистанційного управління, рухомий об'єкт (РО), системи дистанційного управління.

Обсяг: 57 сторінок., 1 таблиця, 19 рисунків, 4 додатки, 17 джерел.

**Kolomiiets Mariia. Remote control system on the Webots platform with a modular type of behavior and SQL database.** Specialization 122 «Computer science», «Software developing and computer graphics». Vasyl' Stus Donetsk National University, Vinnytsia 2021.

The qualification (bachelor's) work investigates the specifics of simulators, especially Webots. The advantages and disadvantages of existing intelligent solutions in the field of existing control systems for robotic devices are shown. A robotics control system on the Webots platform and a database for the developed application have been designed.

Key words: Programmable Logic Controller (PLC), Programmable Automation Controller (PAC), automated mobile platform, remote control unit, software, software complexes for remote control, mobile object, remote control systems.

Consist of: 57 pages, 1 table, 19 images, 4 additions, 17 sources.

## ЗМІСТ

ВСТУП .....	4
РОЗДІЛ 1. Огляд сучасних підходів та опис предметної області.....	6
1.1. Історичний розвиток систем управління робототехнічними пристроями.....	6
1.2. Опис предметної області.....	8
1.3. Огляд сучасних підходів дослідження керування робототехніки ..	9
РОЗДІЛ 2. Проектування системи управління робототехнікою.....	15
2.1. Концепція та функціональна модель системи .....	15
2.2. Опис логічної структури .....	20
2.3. Структура фізичної моделі.....	26
РОЗДІЛ 3. Опис реалізації системи віддаленого управління робототехнікою .....	30
3.1. Деталізація структури програмного забезпечення та опис програмних модулів.....	30
3.2. Проектування функціоналу і реалізація роботи з базами даних.....	33
3.3. Опис інтерфейсу та роботи програми.....	39
3.4. Тестування спроектованої системи.....	41
ВИСНОВКИ.....	49
СПИСОК ДЖЕРЕЛ ПОСИЛАННЯ .....	50
ДОДАТКИ.....	52
Декларація щодо унікальності текстів роботи	<b>Ошибка! Закладка не определена.</b>

## ВСТУП

Сучасна галузь робототехніки вимагає постійного розвитку штучного інтелекту. Саме алгоритм поведінки робота визначає його продуктивність та призначення. Кожен такий винахід перед початком використання проходить ряд тестувань, де особливо детального вивчення потребує штучний інтелект майбутнього приладу. Тому метою кваліфікаційної роботи стало створити середовище для тестування користувацьких модулів керування поведінкою роботів.

Актуальність теми дослідження зумовлена необхідністю створення автоматизованого рішення оцінки ефективності логіки управління.

Мета – проаналізувати, спроектувати та розробити систему керування робототехнікою. До завдань бакалаврської роботи належать:

- дослідити предметну область і виявити потребу певного функціоналу систем управління роботами;
- спроектувати і розробити ПЗ управління робототехнікою за допомогою об'єктно орієнтованої мови програмування;
- спроектувати реляційну базу даних;
- впровадити функціонал завантаження власних модулів/класів алгоритмів управління;
- протестувати готовий додаток.

Об'єктом дослідження є різноманітні симулятори та платформи керування робототехнічними пристроями, їх архітектура та можливості. Предмет дослідження – функціонал існуючих технологічних рішень в галузі управління.

Теоретичне та практичне значення одержаних результатів полягає в подальшому практичному використанні при проектуванні власних інтелектуальних рішень в області керування.

Матеріали кваліфікаційної (бакалаврської) роботи було апробовано на II Всеукраїнській науково-практичній конференції «Прикладні інформаційні



технології» 29 квітня 2021 р., де була представлена доповідь на тему: «Система віддаленого управління робототехнікою на платформі Webots з модульним типом поведінки та SQL базою даних.». Також результати викладено в статті: «Система віддаленого управління робототехнікою з модульним типом поведінки на платформі Webots» у «Віснику Студентського наукового товариства ДонНУ імені Василя Стуса» в травні 2021 р.

Окрім вище зазначених статей, на основі аналізу предметної області, отримані матеріали були оформленні у вигляді тез та опубліковані в збірнику конференції «Всеукраїнської науково - практичної конференції «КТОД» 5 грудня 2020 р.: «Система віддаленого управління робототехнікою з модульним типом поведінки на платформі Webots».

Також, взято участь у 1-му етапі конкурсу Всеукраїнських наукових студентських робіт в січні 2021 році. Тема наукової роботи: «Система віддаленого управління робототехнікою з модульним типом поведінки на платформі Webots», шифр: «Дистанційне управління робототехнікою». В результаті конкурсу було зайнято 2 місце.

Структура кваліфікаційної (бакалаврської) роботи передбачає такі структурні підрозділи: аналіз предметної області, проектування ПЗ і його реалізація.

Аналіз предметної області описується в підрозділах: історичний розвиток систем управління робототехнічними пристроями; опис предметної області; огляд сучасних підходів дослідження керування робототехніки. Детальний огляд проектування ПЗ пояснюється в підрозділах: концепція та функціональна модель системи; опис логічної структури; структура фізичної моделі. Опис реалізації системи демонструється в наступних структурних підрозділах: деталізація структури програмного забезпечення та опис програмних модулів; проектування функціоналу і реалізація роботи з базами даних; опис інтерфейсу та роботи програми; тестування спроектованої системи.

## РОЗДІЛ 1. Огляд сучасних підходів та опис предметної області.

### 1.1. Історичний розвиток систем управління робототехнічними пристроями

Перші спроби створення роботизованих пристроїв були ще в античну епоху – створення механічних голубів, постаментів що рухаються, водяні автомати, годинни-клепсидра. Потім в історії робототехніки відзначився відомий Леонардо да Вінчі, який розробляв різноманітні автоматизовані механічні апарати. Наприклад лев, який сам відкривав герб Франції при в'їзді короля в місто. В 18 столітті годинникар П. Жаке-Дроз створив механічну ляльку, яка була запрограмована за допомогою барабанів на написання текстових повідомлень до 40 букв. Проте більш серйозного інтересу роботизована техніка набула в 20 столітті, коли відбувся стрімкий розвиток автоматизованих обчислюваних машин. Приблизно в 30-х роках 20 століття з'являються андроїди які реалізують елементарні рухи і здатні вимовляти по команді людини прості фрази. Саме така технологія була спроектована інженером Д. Уекслі, створена для виставки в Нью-Йорку в 1927 році. В 50-х роках XX століття з'явилися механічні маніпулятори для роботи з радіоактивним матеріалом. Системою управління для такого винаходу були розробки дистанційного керування колісними платформами за допомогою маніпуляторів, телекамери чи мікрофону.

Широке впровадження промислових станків з числовим програмним управлінням стало поштовхом для створення програмуючих маніпуляторів. Суть програмного забезпечення полягала в тому, що оператор за допомогою програматорів вводив керуючу програму в контролер. Команди розміщуються в оперативній пам'яті, а в процесі створення чи редагування керуючої програми, оператор використовував системну програму редактора. За допомогою неї на дисплей виводилися необхідні частини управління в які вносились зміни. Інструкції зчитувались спеціалізованою мовою програмування, наприклад G-код, який після цього інтерпритатором системи переводився із вхідної в команди

управління головним приводом, приводом подач, контролерами управління вузлами станка [1].

В 1954 році американський інженер запатентував метод управління погрузочним маніпулятором за допомогою змінних перфокарт. Згодом у 1956 році у співпраці з Д. Енгельбергом була створена перша промислова компанія «Юнімейшн» по виробництву промислової робототехніки. Це зумовило появу мікропроцесорних систем управління, які в 1970-х роках замінили спеціалізовані блоки управління роботами на контролери які здатні програмуватись.

Надалі жорстка прив'язка програмного забезпечення та фізичного робототехнічного пристрою послаблювалась, впроваджувались мобільні рішення управління робототехнікою, розроблялись технології моделювання «життєдіяльності» роботів.

Різноманітність тенденцій розвитку робототехнічних пристроїв, їх типів, а також вдосконалення програмного забезпечення роблять використання «інтелектуальних машин» привабливим для все більшого числа різних компаній і галузей промисловості. Роботи знаходять застосування в сфері виробництва електроніки, в логістиці і складуванні, у будівництві, дослідженнях, медицині, для видобутку корисних копалин, для транспортування як в космосі так і на землі, в якості іграшок і в багатьох інших сферах.

Контролери роботів зменшуються в габаритах і стають більш доступними для інженерів і розробників. На сьогоднішній день програмовані логічні контролери (Programmable Logic Controller, PLC), контролери для автоматизації (Programmable Automation Controller, PAC), промислові комп'ютери та модульні контролери, які випускаються сторонніми виробниками, можуть використовуватися для керування роботами [1].

Компанії розробки програмного забезпечення постають перед задачею тестування спроектованого контролеру для його подальшого впровадження. Тестування на реальних прототипах роботів не є раціональним підходом, так як потребує значних матеріальних затрат та не менш цінного ресурсу – часу.



Вирішенням задачі оптимізації процесу стало використання симуляторів. Комп'ютерні середовища моделювання дозволяють детально зімітувати поведінку робота і його оточення. Це дає можливість провести повне тестування виконавчих пристроїв (інструментів і маніпуляторів), елементів безпеки в критичних ситуаціях, виконання завдань і місій в спеціальних умовах, яке б засвідчувало правильність проектування і функціонування робота ще до його впровадження або придбання. Яскравими прикладами таких середовищ стали: система моделювання Webots, програма з широким набором бібліотек Gazebo, інструмент програмування робота в віртуальному середовищі Sobot Rimulator, та багатопоточний симулятор – ARGoS [2].

## 1.2. Опис предметної області

Одним з найпопулярніших додатків для симуляторів робототехніки є 3D-моделювання та рендеринг робота і його середовища. Цей тип програмного забезпечення для робототехніки має симулятор, який є віртуальним роботом, що здатний імітувати рух в реальному часі в середовищі оболонки. Деякі робототехнічні тренажери використовують фізичний механізм для більш реалістичної генерації руху робота. Використання симулятора робототехніки для розробки програми управління роботами рекомендується незалежно від того, чи доступний справжній робот чи ні. Симулятор дозволяє зручно писати і налагоджувати в автономному режимі з остаточною версією програми, перевіреної на реальному роботі. Це перш за все стосується промислових роботизованих додатків [2].

Дії робота на основі датчиків набагато складніше імітувати і програмувати в автономному режимі, оскільки рух робота залежить від миттєвих показників датчика в реальному світі. Сучасні симулятори, як правило, забезпечують наступні можливості:

- Швидке прототипування роботів



- Використання власного емулятора як інструмент точного відтворення навколишнього середовища (платформа віртуального роботизованого експерименту, Webots, R-Station, Marilou, 4DV-Sim).
- Використання зовнішніх інструментів.

Більшість симуляторів використовують ODE або PhysX - фізичні механізми обробки інформації для реалістичних рухів. До таких симуляторів належать Gazebo, LpzRobots, Marilou, Webots, Microsoft Robotics Studio, 4DV-Sim. Для створення середовищ можна використовувати стандартні інструменти 3D-моделювання або інструменти сторонніх розробників, а також динамічні тіла роботів із вбудованими сценаріями [2].

Серед новітніх технологій, доступних сьогодні для програмування, є ті, які використовують віртуальну симуляцію. Моделювання з використанням віртуальних моделей робочого середовища і самих роботів може надавати переваги як для комерційної компанії, так і для окремого програміста. Використовуючи симуляцію, витрати знижуються, а роботи можуть бути запрограмовані автономно, що виключає час простою. Дії робота і деталі збірки можуть бути візуалізовані в тривимірній віртуальному середовищі за кілька місяців до того, як будуть створені навіть прототипи. Написання коду для моделювання також простіше, ніж писати код для фізичного робота. Хоча перехід до віртуальних симуляцій для програмування роботів є кроком вперед в плані гнучкості користувацького інтерфейсу, багато такі додатки знаходяться тільки в початковому стані або є складними для налаштувань і масштабування.

### 1.3. Огляд сучасних підходів дослідження керування робототехніки

Розглянемо існуючі технічні рішення симуляторів робототехніки аби всебічно описати предметну область додатку що розробляється. Webots - це середовище розробки, використовуване для моделювання, програмування та симуляції мобільних роботів, яке часто використовується в навчальних цілях.

Система моделювання Webots є кросплатформним (Windows, Mac і Linux) симулятором роботів з безліччю можливих інтеграцій: MATLAB, ROS (Robotic Operating System) і API для мов програмування C ++, Java і Python. Webots надає велику бібліотеку моделей сенсорів, інтерактивну 3D-візуалізацію, підтримку CAD систем, набір роботів і оточень, бібліотеку для комунікацій між роботами і виведення даних в спеціальні вікна. Серед наявних бібліотек є найбільш використовувані сенсори і виконавчі пристрої: датчики світла, акселерометри, камери і деякі інші [3].

Програма базується на фізичному механізмі Open Dynamics Engine та механізмі рендерингу OpenGL. Він випущений за ліцензією Apache 2.0.

Головними характеристиками даного ПЗ є можливість взаємодіяти з моделлю управління під час моделювання, а також моделювати власні зразки і імпортувати їх в середовище. Webots використовує ODE (Відкритий динамічний двигун) для виявлення зіткнень та динамічного моделювання твердого тіла. Іншими словами, вбудовано функціонал імітації фізики об'єктів. Основний елемент розробки в Webots - це, так звана, сцена. Вона складається з різних об'єктів оточення, налаштувань фізичних параметрів віртуального світу, а також роботів. Налаштування фізичних параметрів досить гнучкі, вони дозволяють вивчати поведінку робота при різних умовах. Наприклад, можна змінювати силу тертя, змінювати стан середовища - емулювати водні об'єкти, додавати вітер і т.д. Для проектування сцени існує вбудований редактор, він дозволяє проектувати сцени з різними геометричними об'єктами, а також створювати унікальні моделі роботів. Важливою особливістю Webots є можливість прискореної емуляції, що дозволяє за короткий термін зібрати важливу інформацію про поведінку робота при різних умовах. Вбудоване середовище розробки дозволяє створювати контролери робота на різних мовах програмування (C ++, Java і т.д.) [3].

Gazebo це програма і набір бібліотек, що дозволяють моделювати поведінку роботів з урахуванням фізичних ефектів. Модель робота описується на спеціальній мові SDF або URDF, яка походить від XML.

Набір роботів і модель оточення завантажуються в систему моделювання, після чого можна починати симуляцію. Ініціалізація завжди відбувається в 3D-світі. Підтримується режим як з графічним відображенням об'єктів, так і без нього для прискорення обробки. Основою Gazebo є фізичний механізм обробки, який можна при необхідності переключити перед початком симуляції. За замовчуванням використовується ODE. Стандартні засоби Gazebo надають можливість моделювати силу тяжіння, деякі види тертя, динаміку руху твердого тіла і деякі інші. Можливості середовища можна розширити, використовуючи готові або написавши власні плагіни. З їх допомогою можна як змінити так і додати власний фізичний ефект, або ввести специфічні правила поведінки об'єктів: створити зовнішньо-контрольовані актуатори робота або датчики. Прикладами можливостей цих розширень є інфрачервоні далекоміри, відеокамера, сервомотори, керовані зовнішніми програмами, вбудований в систему моделювання контролер і навіть фізичні ефекти рідини [4].

Незважаючи на ці численні можливості, Gazebo не дуже підходить для симуляції систем з великою кількістю агентів через досить велику ресурсоемність інформації, що обробляється, тому моделювати роботу систем з більш ніж 1 000 агентів важко.

ARGoS – це багатопотоковий симулятор, що орієнтується на підтримку великої кількості роботів в різних фізичних середовищах в рамках однієї симуляції. Він підтримує як 2D-, так і 3D-моделювання. Велика увага в ньому приділена модульності, в результаті чого надається можливість використовувати розроблений код як в симуляції, так і на реальних роботах. Втілена можливість вибирати між різними візуалізаціями і фізичними механізмами обробки інформації, користуватися прискоренням обчислень за рахунок паралельного виконання розрахунків з декількома типами синхронізації і навіть поєднувати кілька різних бібліотек фізичної симуляції в одній моделі світу. Крім цього, ARGoS надає різні типи комунікації між роботами: RFID-мітки, WiFi і т.д.

ARGoS доступний у вигляді вихідних кодів (під ліцензією MIT), а також готових пакетів під MacOSX і декількох дистрибутивів GNU / Linux [4].



Є деяка підтримка Windows. Система підтримує призначені для користувача розширення. Швидкість симуляції може досягати 40% прискорення в порівнянні з реальним часом для 10 000 роботів. Однак, як і багато інших симуляторів, ARGoS не надає особливої підтримки механізмам соціальної взаємодії в колективах роботів, а його масштабованість, судячи з усього, обмежується використанням локальних ресурсів.

Основні характеристики розглянутих систем відображені в таблиці 1.2.1.

Таблиця 1.2.1 – Порівняльна таблиця основних характеристик систем.

Сист. Моделюван- ня	Імітація фіз. Властивостей об'єкта	Реаліз. бібліотек елементів робота	Підтри- мка комуні- кації між аген- тами	Підтри- мка великої кільк. агентів	Рі- вень під- три- мки	Підт- римка обчис- лень на клас- терах	Наяв. Соціа- льно орієнто- ваних бібліотек
Webots	+	+	+	-	+	-	-
Gazebo	+	+	-	-	+	-	-
ARGoS	+	+	+	+	+	-	-

Ці критерії розшифровуються наступним чином:

- імітація фізичних властивостей об'єкта - моделювання фізичних явищ, механічні обмеження (тертя, в'язкість, зв'язок між частинами об'єкта і т.д.);
- реалізація бібліотек елементів робота - набір готових реалізацій сенсорів і актуаторів;
- наявність підтримки комунікації між агентами - моделювання спілкування (зв'язку) між агентами і надання стандартного інтерфейсу, визначеного протоколу і повідомлень;
- підтримка великої кількості агентів - пристосованість системи для роботи з великими (близько 10 000) групами роботів;

- рівень підтримки - сумарна оцінка якості документації користувача, рівня підтримки з боку спільноти і розробників, інтуїтивність роботи з системою, документованість коду (в разі відкритого коду);
- підтримка обчислень на кластерах - вбудована можливість роботи системи на кластері;
- наявність соціально орієнтованих бібліотек - надання вбудованих або призначених для користувача бібліотек, що реалізують хоча б деякі з соціальних механізмів.

Порівняльний аналіз наведених програмних продуктів показує, що вони лише частково задовольняють специфіці моделювання багатоагентних робототехнічних систем і штучних колективів з соціальною структурою, і дозволяє скласти більш точні вимоги до такого середовища моделювання [5].

ПЗ моделювання колективів має досить обмежений набір можливостей і функціонал масштабування. Незважаючи на універсальність і потужність, в середовищах розробки і бібліотеках імітаційного моделювання не вистачає хоча б деяких з наступних елементів:

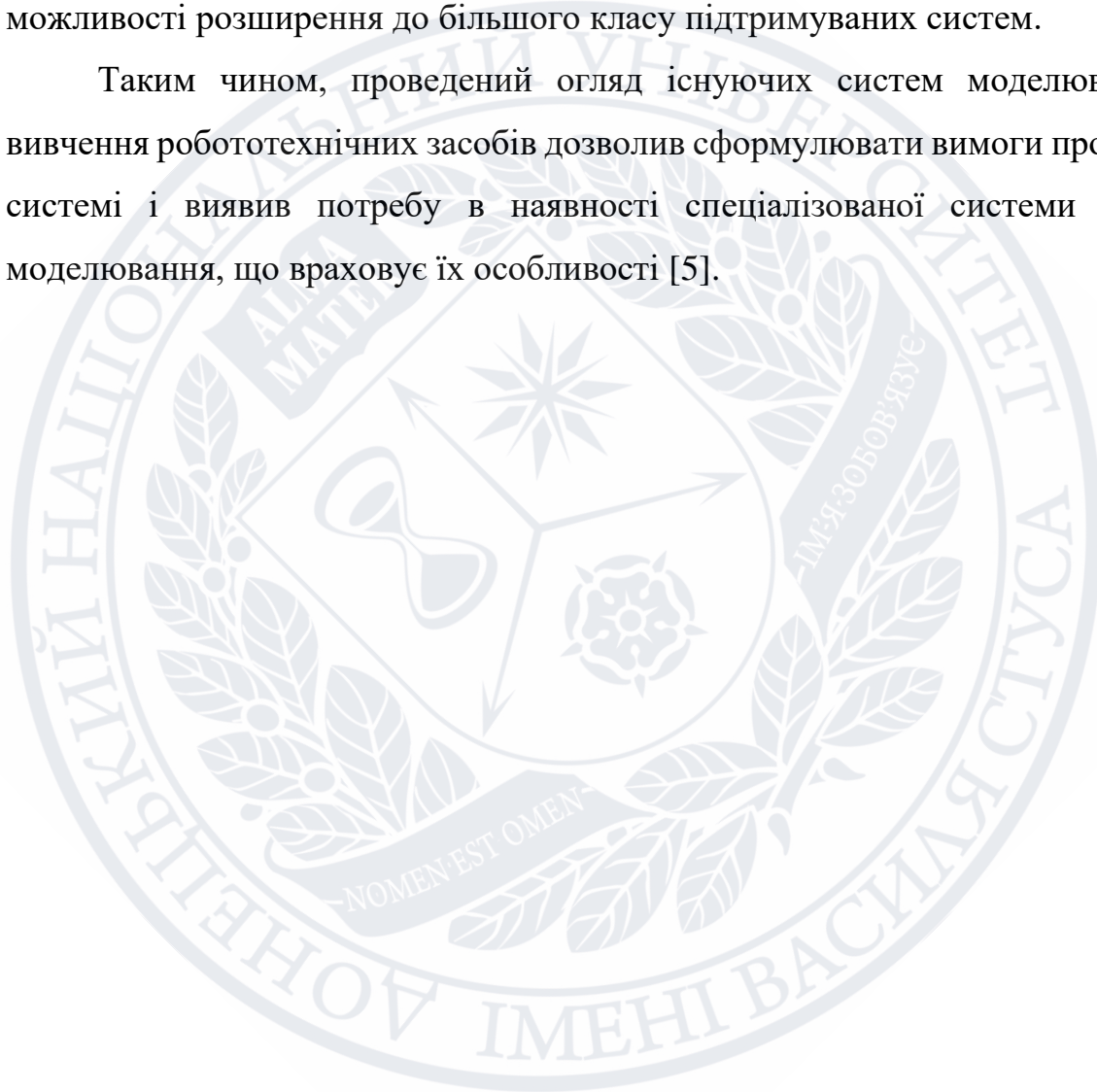
- підтримка локальної комунікації
- алгоритми формування коаліцій
- підтримка просторових і часових відносин.

ПЗ моделювання роботів більше підходить для окремих роботів і фізичних ефектів, а не для систем з великою кількістю агентів або має ті ж проблеми, що і системи імітаційного моделювання: спеціальні засоби для моделювання симуляцій є дещо стандартизовані й одноманітними, що означає необхідність досліднику самотійно реалізовувати численні механізми, алгоритми поведінки. Отже виділимо основні технічні вимоги, які є важливими для використання системи різними групами дослідників:

- надання інструментів або інтерфейсів для інтеграції коду в сторонній, призначений для користувача графічний інтерфейс і візуалізації даних, а також базового варіанту інтерфейсу;

- експорт даних в загальноприйняті, відкриті формати (CSV, GraphML і т.п.), що дозволяють проектувати інтеграцію з іншими інструментами;
- відкритість коду для читання (Open source) дотримання стилів і наявність хорошої документації: добре описане API, архітектура, концепція;
- відкритість коду для модифікації, підтримка плагінів або інших стандартних способів розширення функціональності для виправлення помилок і можливості розширення до більшого класу підтримуваних систем.

Таким чином, проведений огляд існуючих систем моделювання для вивчення робототехнічних засобів дозволив сформулювати вимоги проектованої системи і виявив потребу в наявності спеціалізованої системи агентного моделювання, що враховує їх особливості [5].





## РОЗДІЛ 2. Проектування системи управління робототехнікою

### 2.1. Концепція та функціональна модель системи

У даній роботі вирішується задача автоматизації тестування алгоритмів управління мобільним роботом у середовищі симуляції Webots. Головна ідея проєктованого програмного забезпечення, полягає у створенні платформи, яка комунікує з середовищем симуляції за принципом «обмін повідомленнями» та дозволяє віддалене керування роботом за користувацьким алгоритмом управління. В результаті отримуємо систему з інструментами обробки та збереження інформації від робота та його оточення, засобами роботи з графічною обробкою даних та функціоналом вбудовування власних модулів керування. Одна з головних функцій програмного додатку, полягає в тому, що користувач може протестувати окремо розроблений власний контролер у середовищі симуляції Webots. При цьому користувач через графічний віконний інтерфейс обирає сторонній модуль управління, налаштовує його параметри та слідкує за показниками його відпрацювання [6].

Результатом виконання програми є різноманітні показники роботи системи управління роботом, які відображаються у реальному часі і зберігаються до бази даних. Серед таких показників: демонстрація руху робота в реальному часі, траєкторія руху та пройдений шлях, створена алгоритмом карта місцевості, статистичні графіки та ін.

Для створення необхідних умов дистанційного керування роботом в симуляції та отримання і запис даних від робота, програмну складову необхідно розділити на дві частини – та що безпосередньо виконує роль симулятора, та сторона серверу в якому користувач маніпулює даними. Ці частини повинні бути пов'язані між собою протоколом TCP/IP. Саме цей протокол характеризується як механізм з попереднім встановленням з'єднання, що забезпечує надійність потоку даних, дає впевненість у безпомилковості одержуваних даних і усуває дублювання даних. TCP дозволяє регулювати навантаження на мережу, а також

зменшувати час очікування даних під час передачі на великі відстані. Більш того, ТСП гарантує, що отримані дані були відправлені точно в такій же послідовності.

Така точність процесу передачі даних необхідна для подальших розрахунків переміщення робототехнічного пристрою в просторі симуляції.

Для того щоб організувати доступ до дистанційного керування робототехнічним пристроєм програмну складову було розбито на дві частини. Перша – серверна, розрахована на роботу із користувачем і містить в собі логіку управління та методи запису даних. Друга – клієнтська, представляє собою симулятор Webots, дозволяє створити віртуальну модель навколишнього середовища та протестувати у ньому модель поведінки робота. Модуль поведінки вбудовується в робота і має методи лише для виконання команд які надходять від серверу. Принцип спілкування між сервером та клієнтом будується на основі запит/відгук, де сервер надсилає запит на показники сенсорів, а клієнт у відповідь надсилає список числових характеристик датчиків. На основі датчиків, розраховується швидкість та кут повороту робота і ці дані відправляються на сервер. Цикл закінчується на відповіді клієнта Webots про підтвердження зміни положення робота.

Програма управління роботом розроблялась під операційною системою Windows 10. ОС має технологію InstantGo, яка прискорює завантаження та відновлення роботи. Окрім того Windows 10 пропонує багато вбудованих засобів безпеки та надійно захищає пристрій від шкідливих програм. Система дозволяє з легкістю переходити з одного пристрою на інший. Проте обрана операційна система не є обмеженням, так як додаток проектується кросплатформним [6].

Головним необхідним компонентом для виконання програми є ПЗ Webots. Програма Webots являється розробкою компанії Cyberbotics. В Webots закладено дуже чітке моделювання фізики, що дозволяє створювати крокуючі, колісні, літаючі, плавальні роботи, а також модульні системи. Є можливість самостійно задавати параметри роботів або підбирати їх за допомогою еволюційних алгоритмів. Доступно проектування тіла робота, навколишнього середовища та супутніх об'єктів симуляції для відтворення умов існування машини. Комбінація

засобів додатку дає можливість виробляти складну зв'язку програмування, тривимірної графіки і симуляції різних пристроїв і приборів [7].

Відповідно до функціональних вимог проектного додатку необхідно застосувати «зіркову» топологію (рис. 2.1.1). Це зумовлено специфікою комунікації симулятора Webots та ПЗ для користувача. Користувацький додаток виступатиме головною ланкою мережі, на якій проводитимуться обрахунки майбутніх дій робототехнічного пристрою, й надсилатимуться в активну клієнтську сесію, яка запущена у Webots.

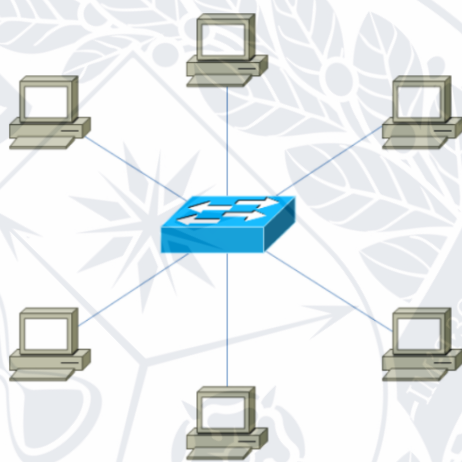


Рис. 2.1.1 – Топологія «зіркової» мережі

Всі сесії симулятора Webots підключаються до центрального вузла, який виступає координатором майбутніх дій робототехнічного пристрою, відповідно який існує в поточній симуляції. Дана модель використовуватиметься в локальних мережах, коли до одного пристрою, що виступає сервером, можуть підключатися кілька різних сесій симулятора. Таку топологію називають «зірка». В такій моделі значно вища відмово-стійкість, а при відключенні, будь-якого клієнту, мережа продовжує спокійно працювати. Однак якщо відмовить центральна ланка, мережа стане непрацездатною. Така логіка комунікації є цілком доцільною, адже передбачає існування декількох сесій в симуляторі, які паралельно обробляються сервером [7].



Представлена, на рисунку, діаграма потоку даних описує зовнішні по відношенню до системи джерела і адресати інформації, логічні операції, потоки і сховища даних, до яких здійснюється доступ. Такий структурний аналіз демонструє, звідки формуються дані для зберігання й момент їх внесення до бази, та процеси для яких необхідно ці відомості дістати із сховища. Для зображення такого процесу обрано нотацію Йордана. Коло означає процес, прямокутник світлого тону – зовнішню сутність, темного – сховище даних, направляючі вказівники – потік даних.

Відповідно до діаграми нижче визначено, коли користувач реєструється в системі на сервері, потік введених даних направляється в базу даних `mySQL`. Під час процесу автентифікації відомості користувачів витягуються із сховища.

Наступний потік даних стосується робота і його показників. Джерелом процесу «Зняття й передача серверу відомостей» виступає об'єкт управління сесії клієнта, потік даних якого направлений в базу даних на збереження. Зворотна процедура – ініціалізація параметрів робототехнічного пристрою на початку програми, з клієнта на сервер. Останній потік, що розглядається, несе вихідні дані обчислень процесу виконання алгоритму управління до бази даних. Таким чином до сховища серверного модулю додається результат виконання програми – команди для формування поведінки робота (Додаток А).

Протокол функціонування полягає в співпраці трьох підрозділів системи: симулятор `Webots` (клієнт), програма користувача(сервер), сторонній модуль управління.

`Webots` представляє собою програму моделювання віртуального середовища та об'єктів в ньому. Початок роботи `Webots` визначається запуском сесії. Наступний крок, це підключення до серверу з яким в подальшому буде відбуватись постійний цикл обміну повідомленнями. Оточення ПЗ зчитує дані з датчиків робота й надсилає отримані показники на сервер. Далі, сесія очікує команд від серверу, в даному випадку це значення швидкості для коліс пристрою. Останнім пунктом циклу роботи клієнтської частини це передання команд на пристрої об'єкту управління (Додаток Д).

Програма користувача несе в собі основні засоби для роботи з симуляцією. Насамперед, це серверна частина цієї системи. Після запуску додатку, сокет серверу очікуватиме з'єднання з клієнтом. В головному вікні програми необхідно ввести параметри користувача та обрати логіку поведінки майбутнього роботу з представлених у бібліотеці (рис. 2.1.2). Після запуску симуляції почнеться цикл обміну повідомленнями між сервером та клієнтом. Перший крок циклу - програма користувача отримуватиме показники з датчиків робота. Далі відомості передаються в головний метод стороннього модулю управління, де визначається подальша поведінка робота. Останній крок циклу – відправлення згенерованих команд симулятору.

Алгоритм обчислення майбутніх показників швидкості коліс повинен базуватись на керуванні роботом з диференціальним приводом. Передбачається, що вхідними параметрами головного методу майбутнього класу, стануть показники сенсорів та датчиків коліс робота, а вихідними – лінійна та кутова швидкості. Скомпільований клас повинен бути імпортований у спеціальну бібліотеку програми [8].



Рис. 2.1.2 - UC-діаграма прецедентів для користувача

## 2.2. Опис логічної структури

Для того щоб розробити додаток відповідно до концепції клієнт-серверного архітектури, в якому відбуватиметься завантаження користувацьких класів управління роботом, збереження/обробка проміжної інформації, було використано технології:

1. Динамічного завантаження Java-класів. Однією з основних особливостей платформи Java є модель динамічного завантаження класів, яка дозволяє завантажувати виконуваний код в JRE без перезавантаження основного додатку. Така особливість отримала останнім часом високу популярність так як широко використовується в серверних ПЗ [8].

Це дає величезні можливості по створенню дійсно модульних систем. Коли інтерфейси і їх реалізації розділені, класи можуть бути завантажені і вивантажені динамічно. Це має сенс, коли потрібно завантажити клас під час виконання програми, коли потрібно замінити клас, змінивши, наприклад, якусь логіку, без перезапуску основного процесу, що в свою чергу дає можливість будувати гнучкі програми.

Реалізація такого процесу буде виконуватись за допомогою пакету ClassLoader. System Classloader - системний завантажувач, реалізований уже на рівні JRE. Цією бібліотекою завантажуються класи, шляхи до яких вказані в змінній оточення CLASSPATH [8].

2. Розділення інтерфейсу, логіки та функціоналу інтеракції з користувачем. Для втілення даної технології було застосовано патерн проектування MVC. Цей патерн використовується щоб розділити основні аспекти додатку (логіку введення, бізнес-логіку і логіку UI), забезпечуючи при цьому вільний зв'язок між ними та мати можливість легко вносити зміни до кожного з них без впливу один на одного.

3. Клієнт-серверну архітектуру за протоколом TCP/IP. Сокети по протоколу TCP / IP служать для реалізації надійних двонапрямлених, потокових з'єднань між хостами в мережі. Сокет може служити для підключення системи



введення-виведення в Java до інших програм, які можуть перебувати як на локальному девайсі, так і на будь-якому іншому в мережі Інтернеті.

У Java підтримуються два різновиди сокетів по протоколу TCP/IP: один - для серверів, інший - для клієнтів. Клас `ServerSocket` служить "приймачем", чекаючи підключення клієнтів перш, ніж зробити якісь дії. Клієнтський служить для підключення до серверних сокетів і ініціювання обміну даними за мережевим протоколом. Клієнтські сокети найчастіше застосовуються в прикладних програмах на Java [9].

4. JDBC технологію взаємодії Java з `mySQL`. JDBC - кросплатформний незалежний стандарт взаємодії Java-додатків з різними СУБД, який реалізований у вигляді пакета `java.sql`, що входить до складу Java SE (Standart Edition). Java Database Connectivity - це стандартний API для незалежного з'єднання мови програмування Java з різними базами даних (далі - БД).

JDBC вирішує наступні завдання: створення з'єднання з БД; створення SQL виразів; виконання SQL запитів; перегляд і модифікація отриманих записів.

Для доступу до кожної конкретної БД необхідний спеціальний JDBC - драйвер, який є адаптером Java - додатку до БД. Драйвери можуть завантажуватися під час роботи програми динамічно. Основними перевагами є: JDBC дозволяє працювати з СУБД без глибоких знань специфіки конкретної бази даних; при зміні СУБД вихідний код програми практично не змінюється, якщо програма була розроблена відповідно до стандарту SQL (без використання відмінних рис попередньої СУБД); не потрібні додаткові установки і настройки програми, яка працює з базою даних; до СУБД можна під'єднатися з використанням URL. [9]

За мову програмування було обрано Java. Насамперед цю мову обрано через її кросплатформність. Незалежність від платформи на якій виконуються програми, це дійсно одна з основних переваг мови, адже один і той же код можна запускати під управлінням операційних систем Windows, Solaris, Linux, Machintosh та ін. Це необхідно, коли програми завантажуються через Інтернет для подальшого виконання під управлінням різних операційних систем. Крім

того, Java - повністю об'єктно-орієнтована мова. Всі сутності в мові Java є об'єктами, за винятком небагатьох основних типів (primitive types), наприклад чисел. (За допомогою Об'єктно-орієнтованого програмування легко розробляти складні проекти.) Java володіє високою надійністю. Творці забезпечили мову Java засобами, що дозволяють виключити саму можливість створювати програми, в яких були б приховані найбільш поширені помилки.

Додаток бакалаврської роботи містить декілька основних модулів які відрізняються призначенням та розташуванням в пакетах програми.

Evaluating package модуль несе математичний зміст та призначений для тригонометричних перетворень. Функціонал модулю використовуються при обчисленнях кутових швидкостей коліс робототехніки, обчислення куту повороту об'єкта, перенесення та зміни систем координат, обчислення векторів.

Модуль CoordinatesEstimation містить інструментарій для роботи з векторами. Найчастіше використовується при обчисленні нового положення робота відносно попереднього, враховуючи показники датчиків і сенсорів та пройдених шлях.

ThirdParty\_Module модуль використовується для впровадження опції імпортування сторонніх модулів управління робототехнікою. ModuleLoader опрацьовує байт код з файлу стороннього модулю та завантажує його в середовище безпосередньо.

Модуль Robot опрацьовує об'єкт робота. Несе інформацію про модель пристрою, технічні параметри, кількість та місце розташування датчиків й сенсорів, відстань між колесами та їх радіус, детальний опис техніки та призначення.

Модуль Data\_Access\_Object призначений роботи з базою даних. Містить методи для маніпуляцій з даними: збереження та читання проміжної інформації симуляції, повідомлень клієнта та сервера, відомостей користувача, технічних характеристик середовища і робототехнічного пристрою, файлів сторонніх модулів управління [9].

View модуль містить функціонал який стосується відображення даних користувачеві. Саме в ньому міститься інструментарій, який управляє демонстрацією вікон, сторінок, повідомлень і т.д. В ньому містяться головні три вікна: авторизація, реєстрація користувача та головне вікно програми.

Модуль Controller містить методи, які виконуються для обробки маніпуляцій користувача з системою. Будь-які дії користувача, спрямовані на зміни моделі, обробляються тут: запуск цикл обміну повідомленнями між клієнтом і сервером, запис користувача в базу даних під час реєстрації чи авторизації, з'єднання/роз'єднання з середовищем симуляції і т.д.

Server модуль установлює й контролює підключення до серверу. Містить в собі об'єкти клієнтського підключення, адреси, домени й відслідковує стан підключення. Тут опрацьовуються всі можливі помилки під час «спілкування» клієнта й сервера та обробка розриву зв'язку між ними [10].

Вхідними даними для програми є показники сенсорів робототехнічного пристрою. Саме ці числові характеристики необхідні для відпрацювання логіки автоматичного та ручного керування пристроєм. Вони надходять від серверу за запитом клієнта. До вхідних даних клієнта належать відомості які вносить безпосередньо сам користувач – логін, пароль, модулі управління логікою робота і т.д.

Вихідні дані записуються в базу даних і являються результатом відпрацювання симуляції. До таких даних належать: момент часу в якому записані відомості, швидкість та кут повороту коліс машини, показники усіх сенсорів і середовища симуляції. Також до вихідних даних належать результати виконання симуляції: час проходження маршруту, карта місцевості визначена як координати точок, траєкторія руху, алгоритмічні розрахунки.

Принцип логіки функціонування полягає в створенні певної екосистеми де співпрацюють три структури: симулятор Webots, програма користувача, сторонній модуль управління [11].

Webots - це програма моделювання віртуального середовища та об'єктів в ньому. Логічна послідовність роботи цієї системи виглядає наступним чином:



запуск сесії та підключення до серверу з яким в подальшому буде відбуватись постійний цикл обміну повідомленнями. Зчитування даних з датчиків робота й надсилання отриманих показників на сервер. Отримання команд від серверу, в даному випадку це значення швидкості для коліс пристрою. Передання команд на пристрої об'єкту управління. Програма користувача несе в собі основні засоби для роботи з симуляцією. Серверна частина цієї системи працює наступним чином: після запуску, сокет серверу очікуватиме з'єднання з клієнтом. Вибір логіки поведінки роботу з представлених у бібліотеці. Запуск симуляції й початок циклу обміну повідомленнями між сервером та клієнтом. Перший крок циклу - програма користувача отримуватиме показники з датчиків робота. Далі відомості передаються в головний метод стороннього модулю управління, де визначається подальша поведінка робота [12].

Третя структура уособлює метод впровадження та роботи із сторонніми модулями управління. Насамперед, імплементація майбутнього контролеру починається з перевірки, чи відповідає він вимогам інтерфейсу що являється шаблоном для таких бібліотек. Далі користувач вказує шлях до модулю управління й обирає файл. Після підтвердження власного вибору, модуль імпортується в ПЗ й стає доступним для симуляції.

В додатку Б представлена діаграма описаного бізнес-процесу функціонування додатку. Фіолетовим кольором визначено вхідні/вихідні точки програми. В синіх прямокутниках визначені основні процеси що відбуваються, а стрілки – перехід між ними. Червоними направляючими вказівниками виділено постійний цикл програми – обмін повідомленнями. Жовта трапеція – процес, який не виконується явно у програмі, а обов'язок за виконання цього етапу лягає безпосередньо на користувача (Додаток Б).

ER діаграма з вказанням направленості потоків показана на рис. 2.2.1, а ER діаграма з визначенням відношення даних зображена на рис. 2.2.2.

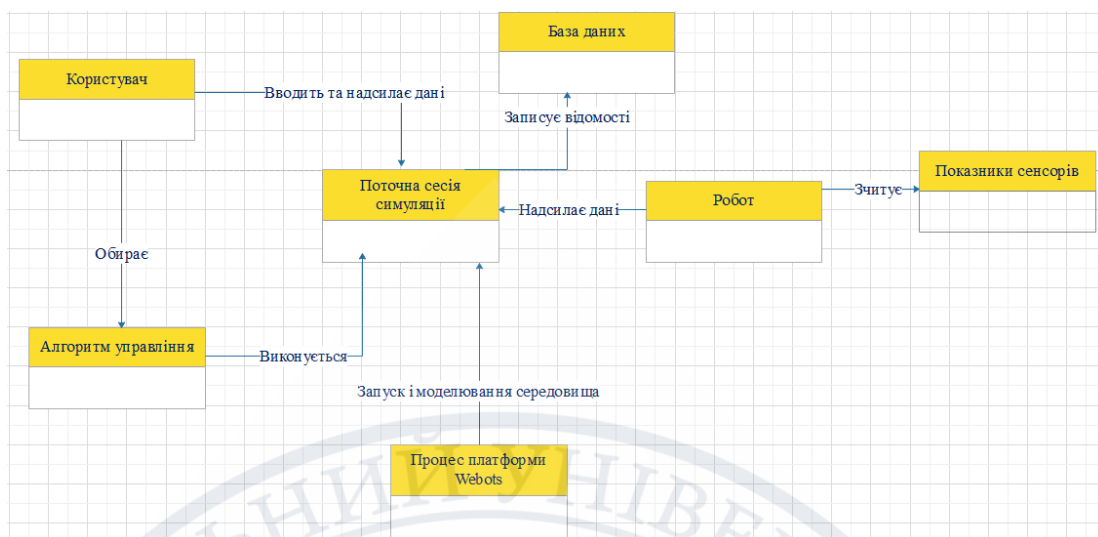


Рис. 2.2.1 ER діаграма з вказанням напрямленості потоків.

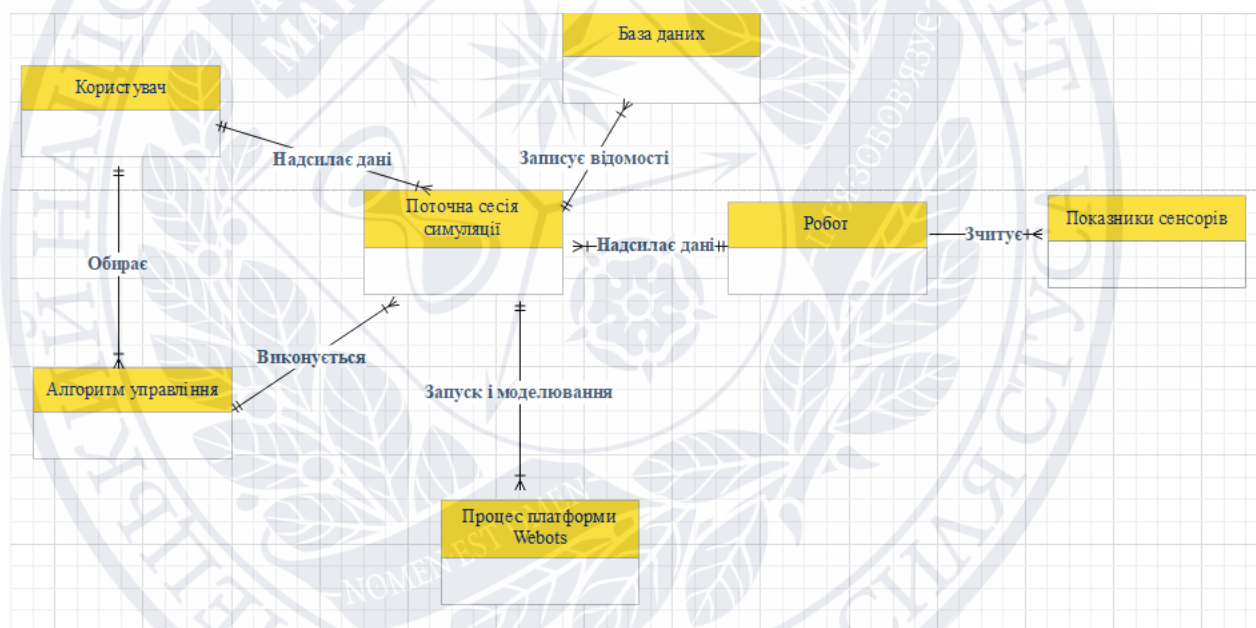


Рис. 2.2.2 ER діаграма з визначенням відношення даних.

В ході аналізу логічної моделі даних, не виявлено проблем відношення чи зберігання даних. Всі атрибути моделей даних, є неключовими та функціонально повно залежать від первинного ключа, тому не вимагається переміщень полів чи створення окремих таблиць (Додаток Е).

### 2.3. Структура фізичної моделі

Найбільш пріоритетним завданням мережевого апаратного забезпечення є надання стабільного, безперебійного й швидкісного зв'язку між сервером та клієнтом. Точність і своєчасність обміну повідомленнями між роботом в симуляторі і середовищем користувача є надзвичайно важливим чинником, адже саме від оперативної реакції сторони клієнта чи сервера залежить подальший розвиток подій та достовірність результатів дослідження користувача.

Отже для нормального функціонування мережі, за протоколом TCP/IP потрібні - мережеві адаптери, концентратори і перемикачі. апаратні і програмні компоненти.

Системою передбачається: два мережевих адаптера Ethernet або Fast Ethernet, концентратор і UTP кабель. Рекомендується мати вбудоване мережеве програмне забезпечення для Windows. Такий набір дозволяє сформувати надійну і продуктивну мережу з мінімумом витрат. При розробці ПЗ використовувались адаптери 10/100 /1000BASE-TX Gigabit Ethernet з роз'ємом для UTP-кабелю та концентратори 10/100 з портами Gigabit Ethernet. Мережевий адаптер повинен підтримувати операції напівдуплексний і повнодуплексний режими. У більшості комп'ютерів мережевий адаптер встановлюється в роз'єм PCI настільних комп'ютерів або слот PC Card в портативних системах [13].

У локальних мережах, для забезпечення надійності зв'язку, потрібно використовувати концентратори. Як правило, він служить для з'єднання кабелів і передачі інформаційних сигналів всіх комп'ютерів мережі. Концентратори повинні використовуватися в мережах з витою парою. Порти концентратора служать точками з'єднання мережевих пристроїв. Комп'ютери та інші пристрої підключаються до концентратора за допомогою окремих кабелів.

Серверна частина ПЗ, що призначена для користувача була розроблена у середовищі IntelliJ IDEA на мові Java. IntelliJ IDEA з точки зору можливостей і ціни поставляється у двох варіантах: безкоштовного Community Edition, і платного Ultimate Edition з розширеною функціональністю. Для проектування



даного ПЗ була обрана Community Edition. Вона призначена для JVM- і Android. IntelliJ IDEA забезпечує єдиний інтерфейс взаємодії з більшістю систем контролю версій, включаючи Git, SVN, Mercurial, CVS, Perforce і TFS. Також IDEA оснащена інструментами для збірки, середовищем виконання тестів, інструментами покриття (coverage tools) і вбудованим 51 термінальним вікном. У IntelliJ немає власного профайлера, але за допомогою плагінів до неї можна під'єднати сторонні. У комплект IDEA входить декомпілятор. Серверне програмування на Java передбачає часте взаємодія з базами даних, так що програмісти IDEA версії Ultimate оцінять зручність інструментів для роботи з SQL і БД. Але якщо комусь їх можливостей буде мало, можна придбати версію IDEA Ultimate з вбудованою SQL IDE (DataGrip). Правда, це буде трохи дорожче, ніж звичайна підписка IDEA Ultimate [13].

IntelliJ IDEA підтримує всі основні сервери додатків JVM, і дозволяє розгортати і проводити налагодження на цих серверах, що нівелює добре знайомі всім програмістам Java Enterprise труднощі. IDEA також підтримує Docker (за допомогою плагіна).

IDEA розширила підтримку коду Spring, Java EE, Grails, Play, Android, GWT, Vaadin, Thymeleaf, Android, React, AngularJS і інших фреймворків.

Для програми управління та аналізу робототехніки обрано середовище симуляції Webots. Webots - програма-симулятор з відкритим вихідним кодом. Широко використовується в промисловості, освіті та наукових дослідженнях. Webots використовує фізичний двигун ODE (Open Dynamics Engine) для моделювання динаміки твердого тіла. Бібліотека ODE дозволяє точно змоделювати фізичні властивості таких об'єктів, як швидкість, інерція і тертя.

Велика колекція вільно модифікованих моделей роботів поставляється з розповсюдженням програмного забезпечення. Крім того, можна створювати нові моделі з нуля. При проектуванні моделі робота користувач визначає як графічні, так і фізичні властивості об'єктів. Графічні властивості включають форму, розміри, положення і орієнтацію, кольори і текстуру об'єкта. До фізичних

властивостей відносяться маса, коефіцієнт тертя, а також пружини і константи демпфування. У програмному забезпеченні є проста динаміка рідини.

Webots включає в себе набір датчиків і приводів, часто використовуваних в робототехнічних експериментах: датчики близькості, датчики світла, сенсори, GPS, акселерометри, камери, випромінювачі та приймачі, серводвигуни (ротаційні та лінійні), датчик положення і сили, світлодіоди, захоплення, гіроскопи, компас і т.д.

Webots надає можливість робити знімки екрану PNG та записувати симуляції як фільми MPEG (Mac / Linux) та AVI (Windows). Webots-світи зберігаються у крос-платформних файлах .wbt, формат яких базується на мові VRML. Також можна імпортувати та експортувати світи або об'єкти у форматі VRML. Інша корисна функція полягає в тому, що користувач може взаємодіяти з запущеним моделюванням в будь-який час, тобто можна переміщати роботів і інший об'єкт за допомогою миші. Програми контролерів роботів можуть бути написані на C, C++, Python, ROS, Java і MATLAB, використовуючи простий API.

Для втілення можливості клієнт серверної архітектури за протоколом TCP/IP використано бібліотеку Java - net.Socket. Для прикладного рішення на Java робота з TCP втілена у взаємодії з об'єктами Socket. Socket - це структура на рівні операційної системи, яка володіє двома головними параметрами: IP-адреса та порт [13].

У процесі обміну, як правило, використовується два сокета - сокет відправника і одержувача. Кожен процес може створити «слухаючий» сокет (серверний сокет) і прив'язати його до якого-небудь порту операційної системи. Хто слухає процес зазвичай знаходиться в циклі очікування, тобто прокидається при появі нового з'єднання. Кожен сокет має свою адресу. Зазвичай клієнт явно «приєднується» до слухача, після чого, під час читання або запису, через його дескриптор передаються дані між ним і сервером. Для доступу до даних, розташованих в системі управління базами даних, додаток використовує JDBC API. JDBC API надає стандартний механізм доступу і управління даними в персистентному сховищі, такому як реляційна база даних. JDBC API дозволяє в

програмі використовувати SQL-команди, які є стандартним засобом доступу до таблиць. Заснований на концепції так званих драйверів, що дозволяють отримувати з'єднання з базою даних по спеціально описаного URL. Драйвери можуть завантажуватися динамічно (під час роботи програми). Завантажившись, драйвер сам реєструє себе і викликається автоматично, коли програма вимагає URL, що містить протокол, за який драйвер відповідає (рис. 2.3.1, рис. 2.3.2)[14].

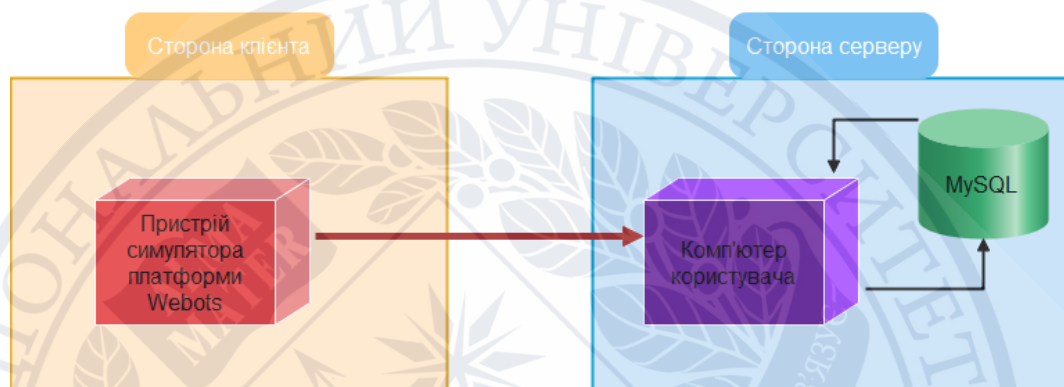


Рис. 2.3.1 Діаграма розгортання

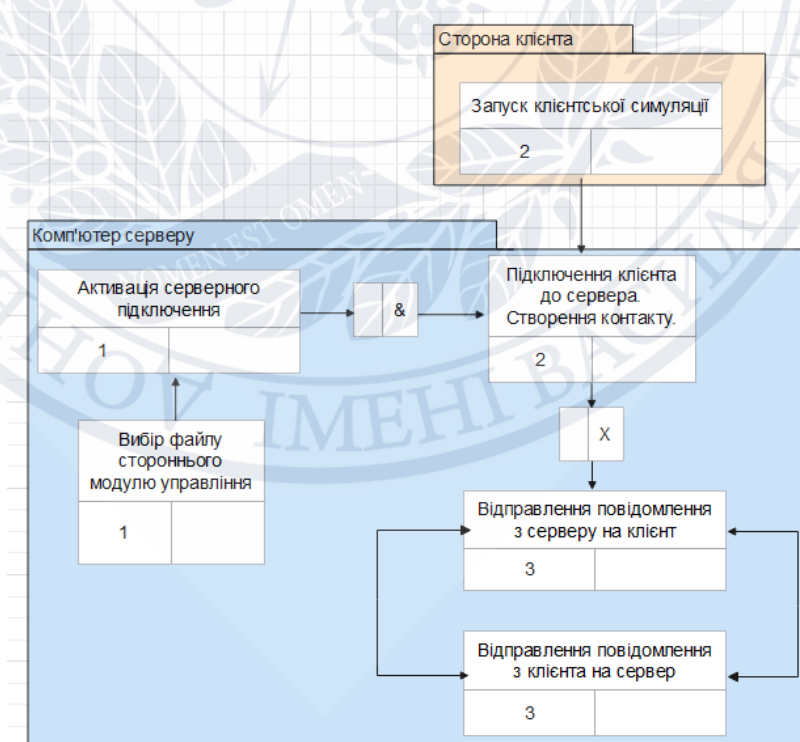


Рис. 2.3.2 Діаграма розгортання



## РОЗДІЛ 3. Опис реалізації системи віддаленого управління робототехнікою

### 3.1. Деталізація структури програмного забезпечення та опис програмних модулів

Програма бакалаврської роботи була написана на мові Java в середовищі IntelliJ IDEA з дотриманням вимог ООП, використовуючи патерни проектування та сторонні бібліотеки.

Об'єктно орієнтований підхід програмування прослідковується в модульній побудові класів, кожен з яких не залежить від іншого, тому легко модифікуватиметься.

В проєкті втілено два патерни проектування: MVC для побудови моделі взаємозв'язку інтерфейсу та функціоналу, та DAO для ефективного управління базою даних. MVC архітектура в структурі програмного проєкту представлена трьома пакетами (package): Model несе в собі логіку додатку, View – форми графічного інтерфейсу, та Controller – функціонал обробки дій користувача. Описана модель продемонстрована на рисунку 3.1.1 [14].

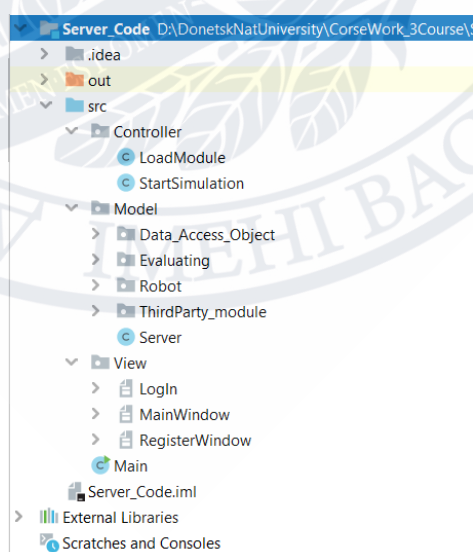


Рис 3.1.1 Структура проєкту на рівні пакетів Model View Controller.

Патерн проектування DAO призначений для абстрагування і інкапсулювання доступу до джерела даних. DAO управляє з'єднанням з базою даних для отримання та запису усіх відомостей програми. В структурі проекту архітектура патерну зберігається в пакеті «Data\_Access\_Object» (рис. 3.1.2). Пакет «domain» містить в собі класи сутностей-об'єктів бази даних. Пакет «mysql» складається з класів які взаємодіють безпосередньо із базою даних, виконують запити додавання, видалення та вставки записів. Пакет «dao» несе інтерфейси та абстрактний клас, який надає базову реалізацію CRUD операцій з використанням JDBC [14].

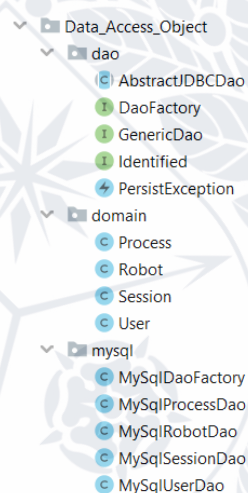


Рис 3.1.2 Структура проекту на рівні пакету Data\_Access\_Object.

Для імплементації інтерфейсу було використано бібліотеку SWING. За допомогою SWING створено три екрані форми: головне вікно додатку і автентифікація та реєстрація користувача. Використано різноманітні компоненти й контейнери, зокрема: JScrollPane, JTextPane, JButton, JLabel, JTextField, JRadioButton, JPanel. Головне вікно складається з трьох робочих зон розділених компонентом JPanel. Перша зона призначення для відображення основної інформації симуляції. Друга для вибору логіки управління роботом. Третя зона визначена для інформаційних повідомлень користувачеві про стан системи.

Для втілення можливості клієнт серверної архітектури за протоколом TCP/IP використано бібліотеку Java - net.Socket. Щоб реалізувати головний інструмент програмного забезпечення – динамічне завантаження сторонніх модулів управління – імпортовано бібліотеки io.File та io.FileInputStream й розширено клас CClassLoader. Аби полегшити математичні обчислення, в клас стандартного набору модулів поведінки програми, імпортовано бібліотеку «Math». Зокрема, у визначенні подальшого руху робота використовувались функції піднесення до степеню, отримання значення косинусу, синусу, тангенсу, арктангенсу [14].

Вхідні дані програмного продукту діляться на три види, тобто мають три різних джерела. Перший потік вхідних даних представлений відомостями, створення яких ініціює користувач. Це може бути введення особистих даних(ім'я, логін, пароль), запуск сесії(тоді в програму увійдуть дані про дату початку сесії, її тип, алгоритм управління), закінчення сесії. Друге джерело вхідних даних виникає коли симуляція запущена і в програму надходять відомості з датчиків, сенсорів робота. Третій тип вхідних даних це скомпільований клас управління робототехнічним пристроєм який вбудовується ззовні. Завантажуючись в додаток, ці відомості обробляються пакетом «ThirdParty\_module», що дає змогу використовувати логіку управління написану сторонніми розробниками.

Виділяється два типи вихідних даних програмного забезпечення. Їх поділ зумовлений різним цільовим призначенням. Перший тип – вихідні дані, що формуються під час роботи циклу обчислення подальшої логіки поведінки робота й представлені набором числових показників швидкостей для кожного колеса. Саме ці дані характеризують нову поведінку контролера. Другий тип, це результат виконання симуляції, що записується у файл. В подальшому, ці дані використовуватимуться для графічного відображення інформації, демонстрації маневрів машини і т.д. [14].



### 3.2. Проектування функціоналу і реалізація роботи з базами даних

Для роботи з базами даних в бакалаврській роботі, було використано стандарт JDBC. JDBC представляє собою опис інтерфейсів і деяких класів, які дозволяють працювати з базами даних для Java. Головним принципом архітектури є універсальний спосіб спілкування з різними базами даних. Самі SQL-запити відрізняються за рахунок різного набору функцій для дат, рядків і т.д. Алгоритм і набір команд для доставки запиту на SQL-сервер і отримання даних від SQL-сервера не відрізняються [15].

Додаток працює з абстракцією JDBC у вигляді набору інтерфейсів. Для кожного типу СУБД використовується своя реалізація (JDBC-драйвер). Система JDBC дозволяє завантажити драйвер для конкретної СУБД і одночасно використовувати компоненти цього драйвера за рахунок того, що до цих компонентів відбувається звернення не безпосередньо, а через інтерфейси. Тобто додаток в принципі не розрізняє, звертається він до Oracle або PostgreSQL - всі звернення йдуть через стандартні інтерфейси, за якими "ховається" реалізація.

У роботі були використані наступні інтерфейси:

- `java.sql.DriverManager`
- `java.sql.Statement`
- `java.sql.PreparedStatement`
- `java.sql.CallableStatement`
- `java.sql.ResultSet`

SQL-запити, використані в розробці програми, можна умовно розділити на дві групи: отримання даних - до них відноситься оператор SELECT; зміна даних - до них відносяться оператори INSERT, UPDATE і DELETE;

Для першої групи використовується метод інтерфейсу `Statement` - `executeQuery()`. Він покриває дуже великий відсоток запитів, наприклад для отримання даних з таблиці. Для другої групи запитів використовується інший метод інтерфейсу `Statement` - `executeUpdate()`. На відміну від `executeQuery()` (який повертає `ResultSet`) цей метод повертає ціле число, яке повідомляє скільки рядків

у таблиці було змінено під час виконання запиту. Інтерфейс `Statement` впроваджений для роботи простих запитів. `PreparedStatement` використаний там, де потрібні параметри з запитом. Запит такого типу дозволяє зробити дві речі: заздалегідь підготувати запит із зазначенням місць, де будуть підставлятися параметри; встановити параметри певного типу і виконати після цього запит з уже встановленими параметрами [15].

Клас `Math` містить методи, пов'язані з геометрією і тригонометрією та іншої математики. Методи реалізовані як `static`, тому можна відразу викликати через `Math.methodName()` без створення екземпляра класу. У класі визначено дві константи типу `double`: `E` і `PI`.

Використано методи для тригонометричних функцій, які приймають параметр типу `double`, що виражає кут в радіанах: `sin(double d)`, `cos(double d)`, `tan(double d)`, `asin(double d)`, `acos(double d)`, `atan(double d)`, `atan2(double y, double x)`; а також методи зведення в ступінь - `pow()`, метод для добування квадратного кореня - `sqrt()` і функцію округлення `abs()`, який повертає абсолютне значення.

Клас `java.lang.Double` обертає значення примітивного типу `double` в об'єкті. Об'єкт типу `Double` містить одне поле з типом `double`.

Для роботи в мережі використовується спеціальний пакет `java.net`, що містить клас `Socket`. Серверний сокет `ServerSocket` містить порт, який буде прив'язаний до серверного сокета. Якщо порт зайнятий або заборонений до використання політикою безпеки комп'ютера, то викликається виняток `IOException`. Значення отримується через виклик функції `getLocalPort()`.

Для роботи з IP-адресами в бібліотеці Java є клас `java.net.InetAddress`, який використовується в конструкторі `ServerSocket`. За допомогою `InetAddress` можна визначити адресу IP локального вузла, а також адреси віддаленого вузла, заданого доменним ім'ям. Нижче представлені найбільш часто використовувані методи серверного сокета [15]:

- `Socket accept()` - очікування підключення клієнта;
- `void bind(SocketAddress endpoint)` - зв'язування `ServerSocket` з певною адресою;

- void close() - закриття сокета;
- ServerSocketChannel getChannel() - отримання об'єкта ServerSocketChannel, пов'язаного з сокетом;
- int getLocalPort() - отримання номера порту, який слухає сокет;
- SocketAddress getLocalSocketAddress() - отримання адреси серверного сокета у вигляді об'єкта SocketAddress;
- boolean isClosed() - перевірка, чи закритий серверний сокет;
- void setReceiveBufferSize (int size) - визначення розміру буфера серверного сокета;

Серверний додаток стартує першим і чекає підключень клієнтів. Після цього стартує клієнтська програма та підключається до сервера. З клієнтського додатку можна відправляти повідомлення, на які сервер повинен відповісти. Сервер створює сокет ServerSocket, відкриває порт і чекає підключень клієнта. Після підключення клієнта сервер формує окремий потік Thread, в який передає порядковий номер клієнта і об'єкт Socket для обміну повідомленнями. Сам сервер продовжує очікувати підключення наступного клієнта [16].

Оскільки програмний продукт бакалаврської роботи проектувався за архітектурою MVC патерну, тому в своїй основі має три пакети Model, View, Controller (рис. 3.2.1).

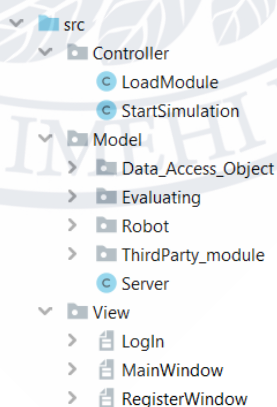


Рис. 3.2.1 Структура проекту на рівні пакетів MVC.



Пакет Model містить всю бізнес-логіку програми та вважається найбільш незалежною від інших. У ньому міститься код, який опрацьовує запити до бази даних, алгоритми управління роботом та впровадження сторонніх модулів управління.

Evaluating package має класи для обрахування прямолінійної та кутової швидкостей роботів. Головний метод класу VelocityEstimation приймає на вхід вектор направлений до цілі та максимальну швидкість, яку може розвивати об'єкт управління. Дані, що повертає функція, є масив дійсних чисел з подвійною точністю(double) - рівномірна прямолінійна швидкість та кутова (рис. 3.2.2).

```
public static double[] get_ang_straight_v_to_target(double[] vector, double maxVelocity){
    double theta = Math.atan2(vector[1],vector[0]); //angular velocity
    double velocity = maxVelocity/Math.pow((Math.abs(theta)+1),0.5);

    return (new double[]{velocity, theta});
}
```

Рис. 3.2.2 Лістинг методу «gtg\_ang\_straight\_v\_to\_target»

Клас CoordinatesEstimation містить методи для роботи з векторами. Суть полягає в отриманні координат початкового положення об'єкту, його цілі та перетворення цих відомостей направлений вектор руху. Клас Additional являється допоміжним для постійних математичних перетворень (округлення числа до знаку).

Пакет ThirdParty\_Module несе в собі логіку по впровадженню сторонніх модулів управління. Інтерфейс Module описує правило створення майбутніх класів управління роботом. Отже контролер повинен містити: метод який би повертав швидкості (радіан в секунду) на кожне колесо окремо – «eval\_to\_wheelspeed», функцію для комунікування із клієнтом – «comunicate\_with\_robot», та метод маніпулювання вхідними параметрами, отриманими від робота, згідно розробленого алгоритму керування – «main\_logic». Клас ModuleLoader наслідує клас ClassLoader, опрацьовує байт код з файлу стороннього модулю та завантажує клас в RunTime(виконуване

середовище). Клас ModuleEngine, зокрема його головний метод створює екземпляр класу логіки керування (рис. 3.2.3)[16].

```
ModuleLoader loader = new ModuleLoader(modulePath, ClassLoader.getSystemClassLoader());

File dir = new File(modulePath);
String[] modules = dir.list();

for (String module: modules) {
    try {
        String moduleName = module.split( regex: ".class")[0];
        if(classModName == moduleName){
            Class clazz = loader.loadClass(moduleName);
            Module execute = (Module) clazz.newInstance();
            return execute;
        }
    }
}
```

Рис. 3.2.3 Лістинг головного методу класу ModuleEngine.

Пакет класів Robot містить об'єкт самого робота. Атрибутами класу Robot\_prototype стали радіус колеса, відстань між колесами, максимальна швидкість, поточна позиція об'єкту управління, координати його цілі й остання зафіксована кутова швидкість. В класі робота визначені методи для руху: обрахування швидкості на основі направлено до цілі вектору («move») й обчислення переміщення пристрою в системі координат («update\_movement\_indicators») (рис. 3.2.4)[17].

```
public double[] move(double velocity, double angular){
    velocity = straight velocity (m/s)
    angular = angular velocity (rad/s)

    double left_wheel = ( (2.0 * velocity) - (angular*wheel_length) ) / (2.0 * wheel_radius);
    double right_wheel = ( (2.0 * velocity) + (angular*wheel_length) ) / (2.0 * wheel_radius);

    return (new double[]){left_wheel, right_wheel}; //rad/s
}
```

Рис. 3.2.4 Лістинг методу «move» класу Robot.

Класи що розміщені в пакеті Data\_Access\_Object призначені для зв'язку з базою даних та управлінням/отриманням записів із сховища. Data\_Access\_Object

складається з трьох компонентів: «dao», «domain», «mysql». «Domain» містить класи що представляють сутності бази у вигляді об'єктів: Process, Robot, Session, User. «Mysql» пакет містить класи для маніпуляцій з даними. На кожну сутність створено окремий такий клас, так як запити, поля, їх кількість суттєво відрізняються. До прикладу, нижче продемонстровано імплементація класу що працює з відношенням «Robot». Метод «getSelectQuery» отримує відомості про усіх роботів з таблиці. «getCreateQuery» - вставляє дані про новий робототехнічний пристрій, а «getDeleteQuery» видаляє запис по ідентифікатору. «getUpdateQuery» отримує відповідний запис, також за ідентифікатором. Пакет «dao» складається з абстрактного класу AbstractJDBCDao та інтерфейсів, які надають базову реалізацію CRUD операцій з використанням JDBC.

Модуль View містить все, що стосується відображення даних користувачеві (рис. 3.2.5). Саме в ньому міститься код, який управляє демонстрацією вікон, сторінок, повідомлень і т.д. Було спроектовано три вікна: авторизації, реєстрації користувача та головне вікно програми.

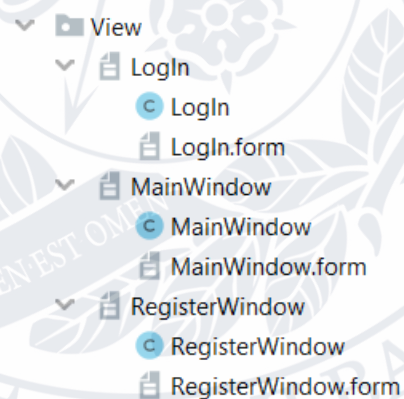


Рис. 3.2.5 Структура пакету «View».

Третя частина Controller містить код, який займається обробкою дій користувача. Будь-які дії користувача, спрямовані на зміни моделі, обробляються тут. При натисканні на кнопку «Start simulation» відпрацьовує відповідний метод цього пакету, який запускає цикл обміну повідомленнями між



клієнтом на сервером та викликає функцію алгоритму обрахування подальших дій робота (рис. 3.2.6) [17].

```
public static void start() throws IOException {  
  
    Server server = new Server( port: 4444 );  
    Go_to_target_logic.go(server.in,server.out,new double[]{100.0,100.0});  
}
```

Рис. 3.2.6 Лістинг методу «start» класу «StartSimulation».

### 3.3. Опис інтерфейсу та роботи програми

Головне вікно програми містить 4 логічні області (рис.3.3.1). Зліва знаходиться блок загальної інформації про сесію симулятора: логін користувача, модель та найменування робота, ідентифікатор сесії та назва алгоритму управління робототехнічним пристроєм. Відомості про користувача заповнюються інформацією за бази даних, які напередодні вносяться під час автентифікації. Поле ідентифікатору сесії генерується системою самостійно та заповнюється автоматично. Модель робота користувач обирає перед початком симуляції в головному меню програми.

Справа знаходиться блок вибору модулю логіки управління роботом. Кнопка “Select logic module” здійснює виклик вікна вибору у файловій системі, аби користувач імпортував необхідний модуль. Після підтвердження вибору, файл відображатиметься в списку “logic list” для подальшого використання в симуляції.

Під блоками загальної інформації та списку імпортованих модулів управління розташовується область запуску й припинення виконання симуляції відповідно.

Область “Status” – призначена для оперативного виведення інформації про поточний стан симуляції що виконується. Під час обміну повідомленнями між

клієнтом і сервером, відомості що вони передають між собою, відображаються користувачу саме в цій текстовій області. Таке рішення було впроваджено для зручнішого аналізу і можливості своєчасного втручання в разі непередбачуваних дій робототехніки. До цієї інформації відноситься: показники датчиків і сенсорів робота, сумарний пройдений шлях, команди сервера на клієнт, кутові швидкості кожного колеса, глобальні показники симуляції (gps, таймер і т.д.).

Рис. 3.3.1 Вигляд головного вікна програми.

Також до забезпечення інтерфейсу програми входять вікна авторизації з полями логіну та пароллю, діалогові вікна для вибору модулю управління, моделі робота, файлу збереження відомостей та їх завантаження для продовження роботи. Також під час початку виконання тестування алгоритму відкривається вікно управління симуляцією з кнопками паузи й відтворення процесу. (Додаток В)

Для того щоб повноцінно використовувати ПЗ необхідно встановити середовище Webots та створити проект робота яким планується керувати в майбутньому. При створенні проекту потрібно обрати модель робота з диференціальним приводом. Рекомендується додавати робототехніку від Khepera. Цей контрольний приклад використовувався при проектуванні додатку, й продемонстрував безперебійну роботу під час тестувань. Створену симуляцію необхідно зберегти у файл та перезавантажити. Далі, після автентифікації, в головному додатку, потрібно обрати логіку керування роботом зі списку встановлених модулів, або із запропонованих стандартних алгоритмів що постачається разом із ПЗ. Після натискання кнопки «Start simulation» запуститься сервер й очікуватиме з'єднання із клієнтом.

Відомо, що в цій системі клієнтом виступає середовище Webots, тому в це час для активації клієнтської частини потрібно запустити симуляцію. Коли сервер й клієнт почнуть обмінюватись повідомленнями – робот розпочне свій рух, що продемонструється в головному вікні симуляції. Після того, як робот виконав усі команди додаток очікуватиме нових вказівок або завершення програми [17].

### 3.4. Тестування спроектованої системи

Тестування спроектованої системи відбувалося наступним чином.

Тест №1: Тестування авторизації

Очікуваний результат: виникнення повідомлення при авторизації з незаповненими усіма полями.

Хід тестування: Крок 1 - Вхід в програму, активація головного вікна авторизації.

Крок 2 – Заповнення поля логіну, поле пароля залишаємо пустим.

Крок 3 - Натиснення на кнопку «authorization»



Фактичний результат: при авторизації користувача з'являється повідомлення про незаповнене поле пароля.

Тест №2: Авторизація користувача в системі.

Очікуваний результат: успішна авторизація користувача, зміна вигляду головного вікна.

Хід тестування: Крок 1 - Вхід в програму, активація головного вікна авторизації.

Крок 2 – Введення логіну “user1” та паролю “6525”.

Крок 3 - Натиснення кнопки «authorization»

Крок 4 – Успішна авторизація, виникнення повідомлення що користувач знайдений в базі, перехід до наступного виду головного вікна.

Фактичний результат: користувач знайдений в базі, авторизація успішна.

Тест №3: зміна паролю користувача.

Очікуваний результат: новий пароль користувача присвоюється, та буде застосовуватись при новій авторизації.

Хід тестування: Крок 1 - В головному вікні авторизації натиснення кнопки змінити пароль.

Крок 2 – Введення логіну “user1”, паролю “1128”

Крок 3 - Підтвердження зміни паролю.

Крок 4 – Авторизація з новим паролем.

Фактичний результат: Авторизація з новим паролем пройшла успішно, що свідчить про присвоєння нового паролю.

Тест №4: імпорт стороннього модулю.

Очікуваний результат: Повідомлення що бібліотека імплементована.

Хід тестування: Крок 1 – Перенесення файлу управління в папку сторонніх модулів.

Крок 2 – Вибір в головному вікні програми файл перенесеної бібліотеки.

Крок 3 - Натиснення кнопки «import».

Фактичний результат: В головному вікні програми з'явилося повідомлення що нова поведінка робота успішно імпортована.

Тест №5: обробка помилки імпорту файлу який не відповідає вимогам програми.

Очікуваний результат: виникнення помилки та повернення програми в попередній стан після імпорту невідповідного файлу.

Хід тестування: Крок 1 - Перенесення файлу управління в папку сторонніх модулів.

Крок 2 - Вибір в головному вікні програми файл перенесеної бібліотеки і натиснення кнопки «import».

Крок 3 - Автоматичний перехід програми на головний екран, виникнення повідомлення що модуль не був імпортований через невідповідність вимогам програми.

Фактичний результат: програма успішно повернулася до попереднього стану після невдалого імпорту, на екран виводиться повідомлення про невідповідність модулю.

Тест №6: поставлення симуляції на паузу.

Очікуваний результат: таймер симуляції призупиняється, в головному вікні Webots робот припиняє рухатись і очікує подальших команд від серверу.

Хід тестування: Крок 1 - Запуск симуляції

Крок 2 - Натиснення на кнопку “Pause”

Фактичний результат: симуляція в програмі Webots стала на паузу, цикл виконання клієнтської частини призупинився. Сервер очікує подальших дій користувача.

Тест №7: зупинка симуляції.

Очікуваний результат: таймер симуляції зупиняється, в головному вікні Webots робот фіксує своє положення в кінцевій точці, клієнтська частина додатку перестає очікувати на повідомлення від серверу й завершує свій цикл роботи. Головне вікно програми повідомляє користувача про успішне закінчення

симуляції, виводить результат виконання дій робота на головний екран у вигляді відомостей пройденого шляху і досягнення пункту призначення [17].

Хід тестування: Крок 1 – Запуск/Відновлення симуляції

Крок 2 - Натиснення кнопки “Stop”

Фактичний результат: симуляція у Webots завершилася. Клієнтська сторона завершила цикл обміну повідомленнями із сервером. Головне вікно програми демонструє відомості про проходження роботом завдання симуляції відповідно до модулю поведінки.

Тест №8: збереження даних симуляції.

Очікуваний результат: запис відомостей симуляції в базу даних.

Хід тестування: запис відомостей до симуляції це автоматичний процес, який відбувається під час циклу обміну повідомленнями між сервером та клієнтом.

Фактичний результат: під час виконання симуляції, відомості робота, його сенсорів та положення записуються в базу даних.

Тест №9: виведення даних попередньої симуляції.

Очікуваний результат: виведення відомостей на головний екран користувача.

Хід тестування: Крок 1 - В головному вікні програми, натиснення кнопки “info”

Крок 2 – вибір з контекстного меню пункт: “simulations data”

Крок 3 – у вікні, що відкрилося, вибір симуляції відповідно до дати та ідентифікатору, та натиснення кнопки “open”

Фактичний результат: на головний екран користувача відкривається csv файл відомостей про відповідну симуляцію.

Тест №10: введення початкових параметрів робота.

Очікуваний результат: заповнення полів назви моделі, радіусу коліс, початкового положення й розташування сенсорів робототехніки.



Хід тестування: Крок 1 – Вибір в головному вікні програми налаштувань робота.

Крок 2 – Заповнення полів радіусу коліс – “5.4”, початкового положення – “1,3,0” й моделі робота – “SM00835”

Крок 3 - Натиснення кнопки “submit”

Крок 4 – Перехід до головного вікна.

Фактичний результат: в базу даних записалися введені відомості про робота.

Тестування сумісності:

Тест №1: тестування в середовищі ОС Windows 10

Очікуваний результат: відсутність вильотів чи виникнення помилок виконання програми через невідповідність платформи операційної системи [18].

Хід тестування: завантаження пакету програмного забезпечення Webots для Windows 10 з мережі інтернет. Інсталяція та запуск проектного додатку, старт симуляції. Встановлення зв'язку між сервером та клієнтом та ініціація обміну повідомленнями між ними.

Фактичний результат: Додаток запустився під ОС Windows 10, сервер й клієнт успішно обмінювались повідомленням. Не було зафіксовано жодного вильоту програми.

Тест №2: тестування в середовищі ОС Linux 64 bit

Очікуваний результат: відсутність вильотів чи виникнення помилок виконання програми через невідповідність платформи операційної системи.

Хід тестування: завантаження пакету програмного забезпечення Webots для ОС Linux 64 bit з мережі інтернет. Інсталяція та запуск проектного додатку, старт симуляції. Встановлення зв'язку між сервером та клієнтом та ініціація обміну повідомленнями між ними.

Фактичний результат: Додаток запустився під ОС Linux 64 bit, сервер й клієнт успішно обмінювались повідомленням. Не було зафіксовано жодного вильоту програми.

### Тест №3: тестування в середовищі ОС Mac OS X 10.14

Очікуваний результат: відсутність вильотів чи виникнення помилок виконання програми через невідповідність платформи операційної системи.

Хід тестування: завантаження пакету програмного забезпечення Webots для ОС Mac OS X 10.14 з мережі інтернет. Інсталяція та запуск проектного додатку, старт симуляції. Встановлення зв'язку між сервером та клієнтом та ініціація обміну повідомленнями між ними.

Фактичний результат: Додаток запустився під ОС Mac OS X 10.14, сервер й клієнт успішно обмінювались повідомленням. Не було зафіксовано жодного вильоту програми.

### Тест №4: тестування в середовищі ОС Mac OS X 10.13

Очікуваний результат: відсутність вильотів чи виникнення помилок виконання програми через невідповідність платформи операційної системи.

Хід тестування: завантаження пакету програмного забезпечення Webots для ОС Mac OS X 10.13 з мережі інтернет. Інсталяція та запуск проектного додатку, старт симуляції. Встановлення зв'язку між сервером та клієнтом та ініціація обміну повідомленнями між ними.

Фактичний результат: Додаток запустився під ОС Mac OS X 10.13, сервер й клієнт успішно обмінювались повідомленням. Не було зафіксовано жодного вильоту програми.

### Тестування безпеки:

Сценарій №1: пароль який вводить користувач в полі головного вікна відображається символом “\*”.

Результат тестування: під час введення паролю користувача під час авторизації чи операції зміни пароля, текст що вводиться відображається за допомогою спеціальних символів.

Сценарій №2: паролі користувачів в базі даних знаходяться в зашифрованому вигляді.

Результат тестування: використовуючи права адміністратора системи було виявлено що паролі до бази даних надходять в зашифрованому вигляді за допомогою алгоритму AES-256.

Сценарій №3: дані, що відносяться до авторизованого користувача, відображаються лише йому і доступ до них має лиш адміністратор системи та безпосередньо сам користувач.

Результат тестування: “user1” авторизувавшись в системі не мав доступу до даних “user2”.

Сценарій №4: лише адміністратор системи з повними правами може змінювати дані безпосередньо використовуючи базу даних.

Результат тестування: після авторизації в системі адміністратора, була спроба змінити дані бази даних – дані успішно оновилися.

Сценарій №5: при зміні пароля користувача системи, старий стає не активним.

Результат тестування: після зміни пароля, користувач спробував авторизуватись під старим паролем – в доступі відмовлено.

Сценарій №6: під час обміну повідомленнями між сервером та клієнтом усі потоки впорядковані, а масив байтів надходять в повному обсязі [19].

Результат тестування: так як зв'язок побудований на TCP/IP протоколі, це виключає можливість неупорядкованості чи неповноти повідомлень.

Сценарій №7: звичайний користувач системи не має прав до зміни даних бази даних

Результат тестування: після авторизації користувача з неповними правами в системі бази даних в доступі було відмовлено.

Тестування зручності використання (юзабіліті):

Тест №1: всі повідомлення про помилки вірні, без орфографічних і граматичних помилок, і відповідають заголовку вікна

Фактичний результат: у діалогових вікнах які сповіщають про помилку введення пароля, логіну, вибору невірної модулю, некоректності значень – повідомлення відповідають заголовку й не мають помилок.



Тест №2: підказки існують для всіх полів

Фактичний результат: під час наведення мишки на поле чи на кнопку, виникає невелике контекстне повідомлення що несе в собі підказку до заповнення чи значення кнопки.

Тест №3: всі тексти правильно вирівняні

Фактичний результат: всі тексти в блоках вирівняні правильно

Тест №4: всі поля правильно вирівняні

Фактичний результат: всі поля в блоках вирівняні правильно

Тест №5: між полями, колонками, рядами і повідомленнями про помилки залишено досить вільного місця;

Фактичний результат: для того аби забезпечити візуальне розмежування в графічному інтерфейсі було витримано відступи між логічними блоками інформації.

Тест №6: всі кнопки повинні мати стандартний формат і розмір

Фактичний результат: кнопки й поля головного вікна масштабовані, ознак нестандартного формату немає.

Тест №7: масштабованість при різних роздільних здатностях екрану (640 x 480, 600x800 і т. д.);

Фактичний результат: під час розтягнення чи звуження головного вікна відстань між елементами вікна пропорційно збільшується/зменшується.

Тест №8: дані у випадаючих списках не обрізуються через розмір поля

Фактичний результат: усі відомості контекстних меню та випадаючих списків не переносяться і не обрізаються. При масштабуванні – змінюють свій розмір відповідно.

Тест №9: гарячі клавіші доступні для управління системою

Фактичний результат: відповідно до інструкцій, користувач має доступ до керування програмою через клавіатуру.

## ВИСНОВКИ

Сьогодення потребує від спеціалістів у галузі автоматики та систем управління широких знань і вмінь у питаннях, що пов'язані з керуванням складними за структурою робототехнікою, виробничими комплексами і системами зокрема.

Програма бакалаврської роботи має на меті розробку програмного забезпечення для контролювання робототехнічним пристроєм та подальшою обробкою даних, отриманих під час функціонування об'єкта. Реалізовується можливість завантаження власним модульм управління поведінкою робота.

Головним завданням стало створення зручного посередника між користувачем й симуляцією.

Отже відповідно до завдань бакалаврської роботи, за допомогою мови Java, бібліотек управління базами даних, модулів створення TCP/IP зв'язку й динамічного завантаження класів, спроектовано систему яка керує моделлю робота віддалено. Таке рішення дозволяє абстрагуватись від специфіки робототехнічних пристроїв та зосередитись на тестуванні ефективності методів управління [20].

Головною функцією системи є динамічне завантаження сторонніх бібліотек, які містять алгоритми управління. Також користувач може зберігати відомості симуляції, робота, своїх сесій та значень приладів, аби мати повне уявлення про свою роботу й виконувати глибший аналіз на основі співставлення даних за різний період.

В подальшому планується розширити набір підтримуваних робототехнічних пристроїв. Наступним кроком буде створення можливості виконання програми в багатопотоковому режимі, що дозволить запускати декілька симуляцій одночасно і оптимізувати процес обчислення.

## СПИСОК ДЖЕРЕЛ ПОСИЛАННЯ

1. «Система віддаленого управління робототехнікою з модульним типом поведінки на платформі Webots» - Коломієць М.В. Стаття у «Віснику Студентського наукового товариства ДонНУ імені Василя Стуса», 2021.
2. «Система віддаленого управління робототехнікою з модульним типом поведінки на платформі Webots» - Коломієць М.В. Доповідь у збірнику матеріалів Всеукраїнської науково - практичної конференції «КТОД», 2020 ст. 71-72.
3. «Система віддаленого управління робототехнікою на платформі Webots з модульним типом поведінки та SQL базою даних» - Коломієць М.В. Наукова робота на II Всеукраїнській науково-практичній конференції «Прикладні інформаційні технології», 29 квітня 2021 р.
4. «Промислові роботи: тренди й типи» [Електронний ресурс]. Режим доступу: <https://www.controleng.com/in/robototekhnika/> - Дата доступу: 11.01.2021
5. Робот [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/Робот> - Дата доступу: 11.02.2021
6. Webots [Електронний ресурс]. Режим доступу: <https://en.wikipedia.org/wiki/Webots> - Дата доступу: 12.01.2021
7. Why webots [Електронний ресурс]. Режим доступу: <http://dspace.tneu.edu.ua/handle/316497/25072> - Дата доступу: 09.01.2021
8. Динамічне завантаження класів [Електронний ресурс]. Режим доступу: <https://habr.com/ru/sandbox/62803/кур> - Дата доступу: 20.01.2021
9. Loading a Java Class at Runtime [Електронний ресурс]. Режим доступу: <https://stackabuse.com/loading-a-jav-runtime> - Дата доступу: 20.01.2021
10. Шаблони проектування - MVC Pattern [Електронний ресурс]. Режим доступу: <https://coderlessons.com/tutorials/java-tekhnologii/shablony-mvc-pattern> - Дата доступу: 20.01.2021



11. Шаблон DAO в JAVA [Електронний ресурс]. Режим доступу: <https://www.codeflow.site/ru/article/java-dao-pattern> - Дата доступу: 15.02.2021
12. Java. Реалізація шаблону DAO [Електронний ресурс]. Режим доступу: <https://www.dokwork.ru/daotalk> - Дата доступу: 15.01.2021
13. Технічні характеристики та вимоги до системи Windows 10 [Електронний ресурс]. Режим доступу: <https://www.microsoft.com/uk-ua/windows10> - Дата доступу: 20.01.2021
14. Розгляд засобів імітаційного моделювання роботів [Електронний ресурс]. Режим доступу: <https://cyberleninka.ru/article/n/obzor-sredstv-imitatsionnogo-modelirovaniya> - Дата доступу: 15.01.2021
15. Java - Networking [Електронний ресурс]. Режим доступу: [https://www.tutorialspoint.com/java/java\\_networking/](https://www.tutorialspoint.com/java/java_networking/) - Дата доступу: 16.02.2021
16. All About Sockets [Електронний ресурс]. Режим доступу: <https://docs.oracle.com/javase/tutorial/networking/sockets/> - Дата доступу: 16.01.2021
17. Webots - програмне забезпечення з відкритим кодом для імітації мобільних роботів [Електронний ресурс]. Режим доступу: <https://ubunlog.com/uk/webots-software-simulacion-robots-moviles/> - Дата доступу: 16.01.2021
18. Microsoft SQL documentation [Електронний ресурс]. Режим доступу: <https://docs.microsoft.com/en-us/sql/> - Дата доступу: 16.01.2021
19. Введення в SQL [Електронний ресурс]. Режим доступу: <https://javarush.ru/groups/posts/1952-vvedenie-v-sql/> - Дата доступу: 16.01.2021
20. Java і бази даних | Підключення до бази даних MySQL [Електронний ресурс]. Режим доступу: <https://metanit.com/java/database/2.2.php> - Дата доступу: 16.02.2021

## ДОДАТКИ

## Додаток А

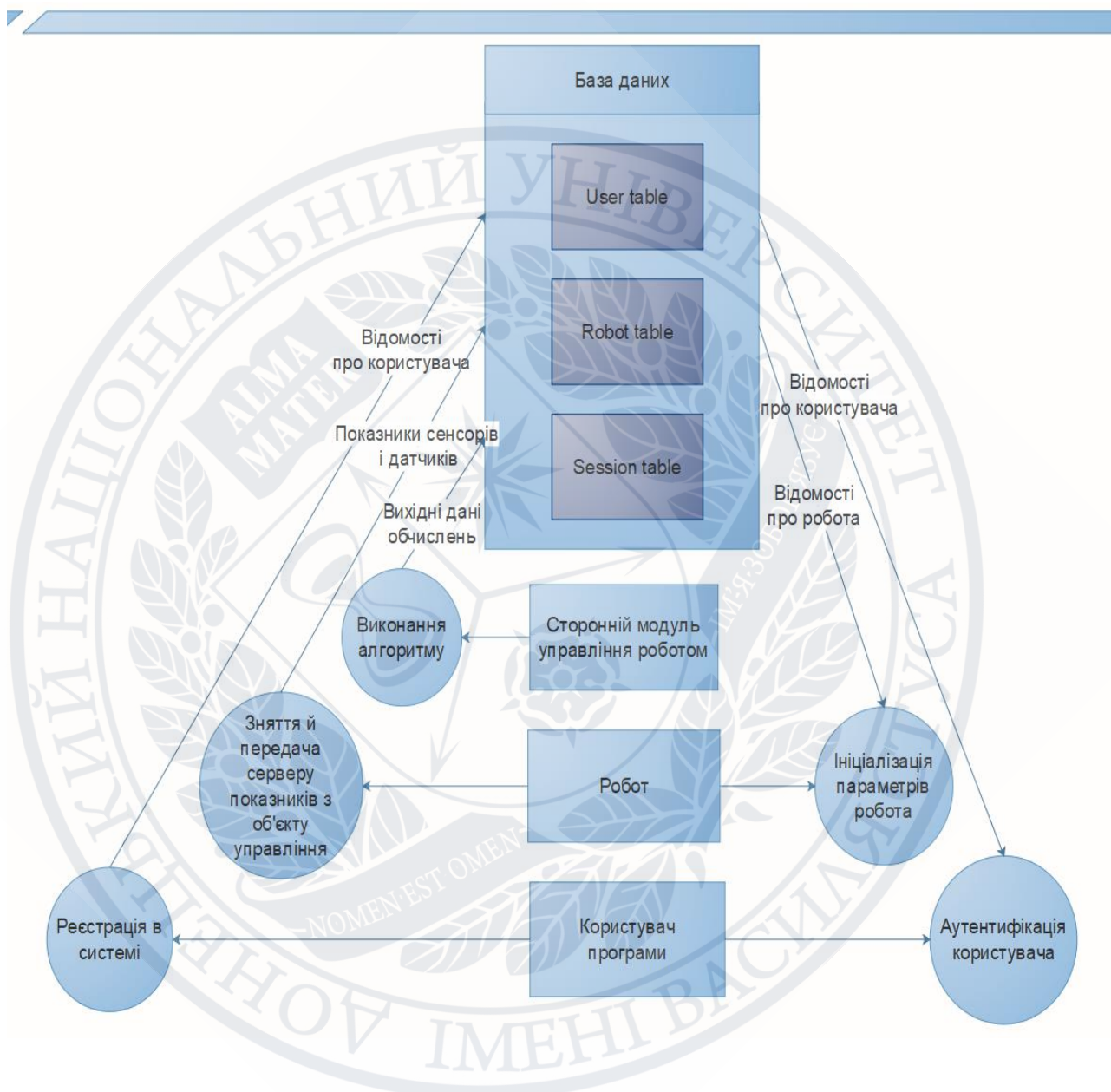


Рисунок - Діаграма потоку даних додатку.

## Додаток Б

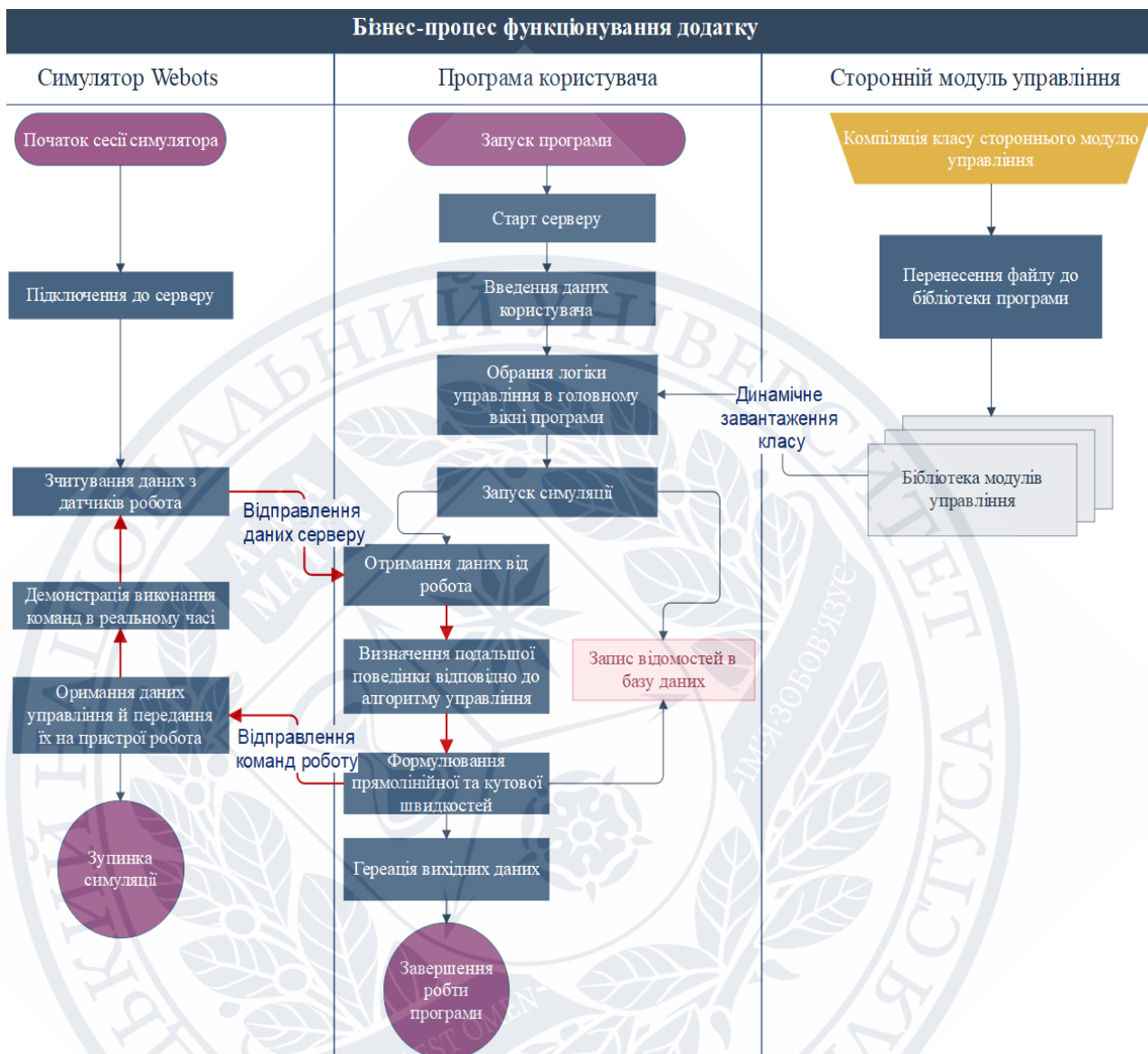


Рисунок - Логічна модель функціонування додатку.



## Додаток В

Registration form	
<div></div>	
User name	<input type="text"/>
Login	<input type="text"/>
Password	<input type="password"/>
<div></div> <input type="submit" value="Submit"/>	

Рисунок 1 - Вікно реєстрації користувача.

Login form	
<div></div>	
Login	<input type="text"/>
Password	<input type="password"/>
<div></div> <input type="submit" value="Submit"/>	

Рисунок 2 - Вікно авторизації користувача.

## Додаток Г

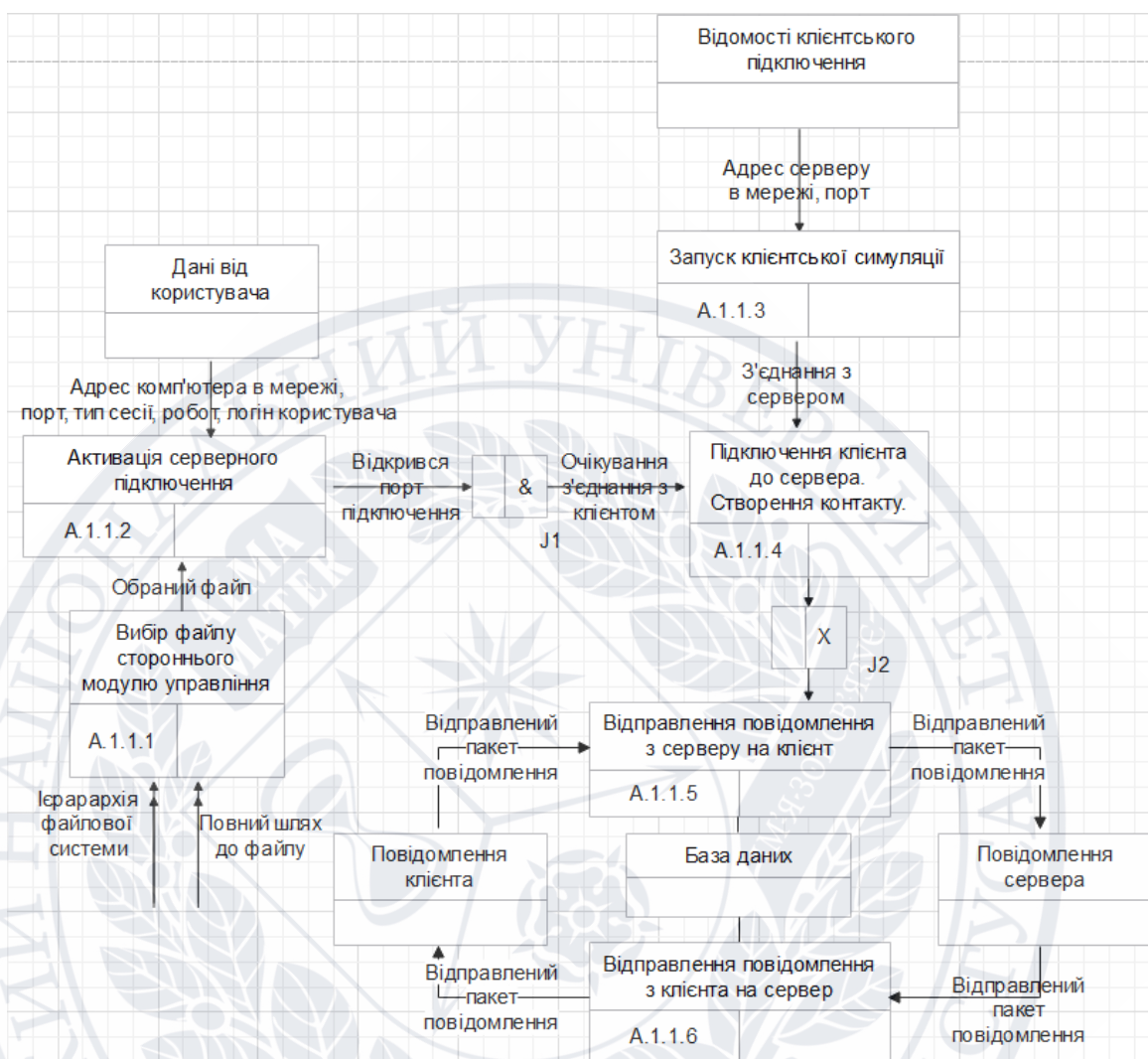


Рисунок - Схема IDEF3 (діаграма PFDD)

## Додаток Д



Рисунок 1- IDEF0-схема блокової моделі системи управління роботом

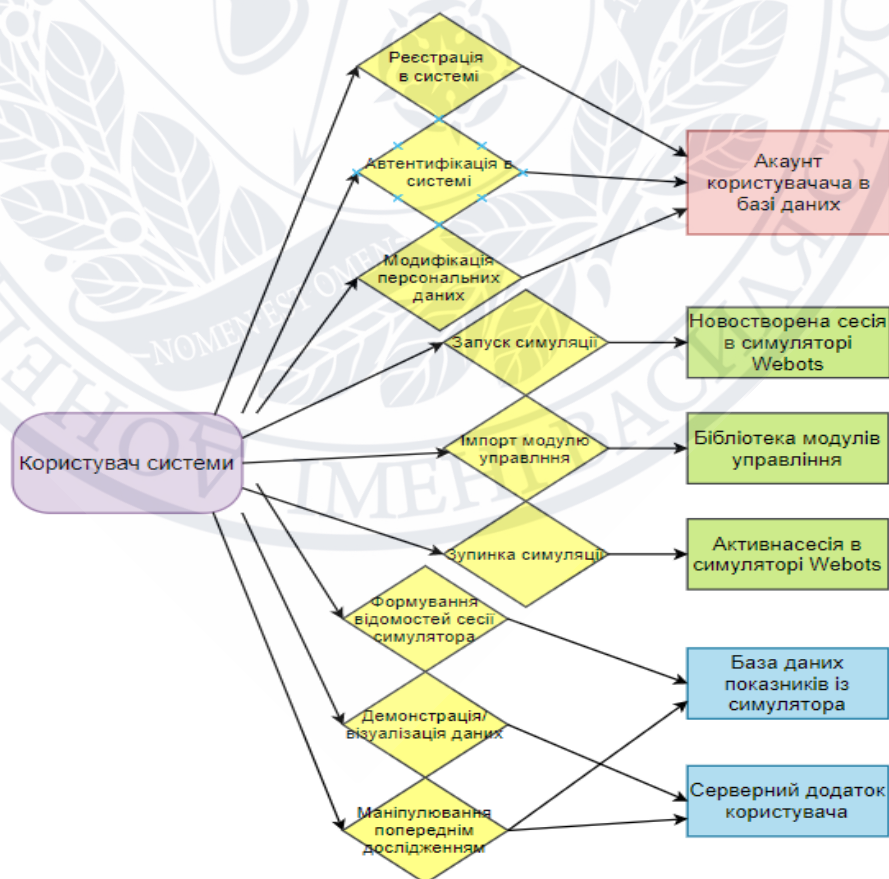


Рисунок 2 - ER-діаграма для користувача системи



## Додаток Е

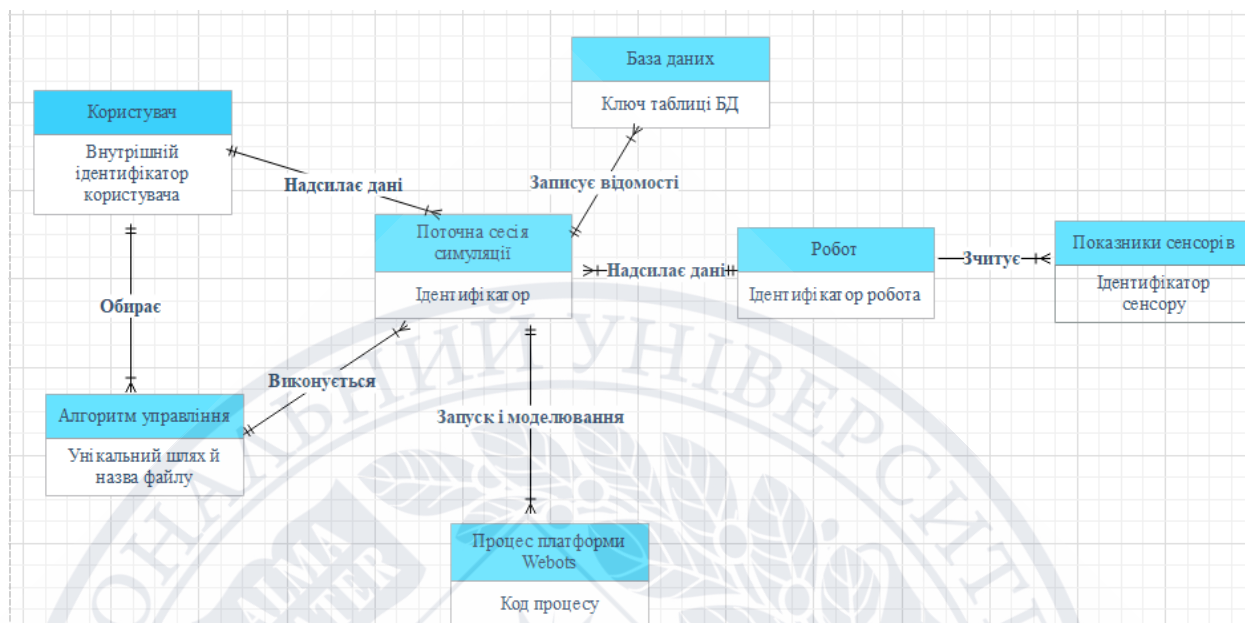


Рисунок 1 - Модель даних, заснована на ключах (Key Based model)

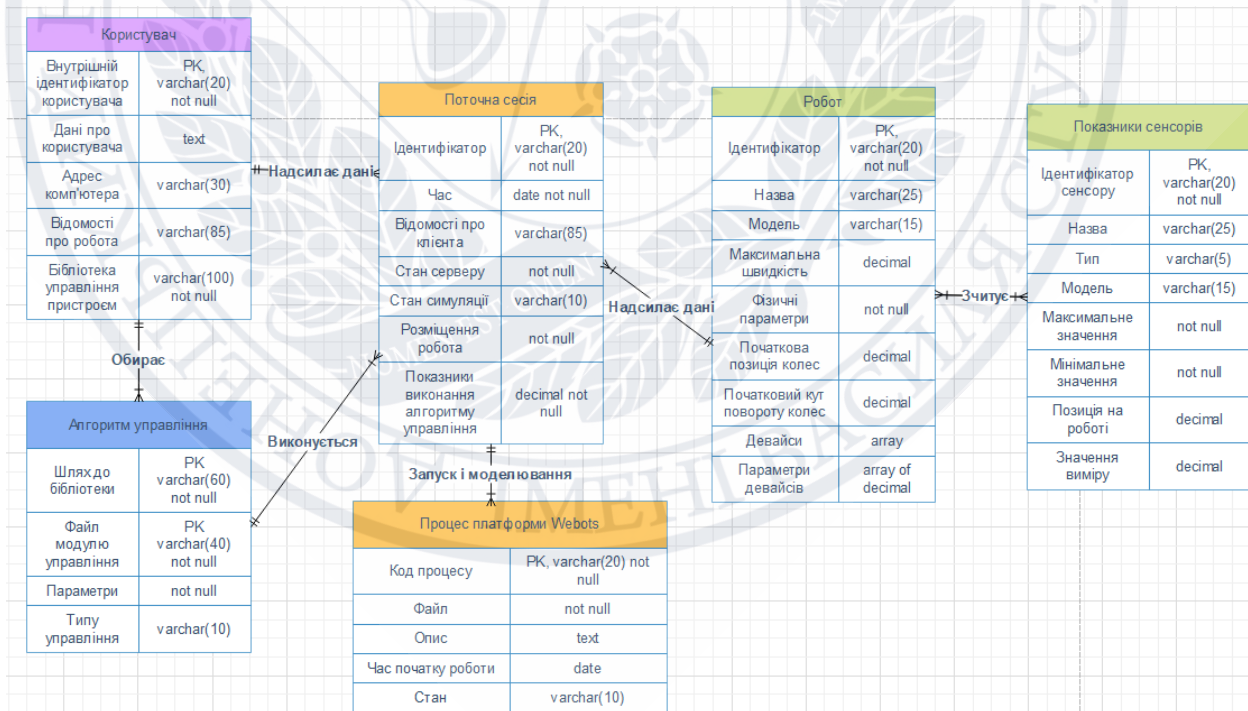


Рисунок 2 - Повна атрибутивна модель (Fully Attributed model)

Декларація щодо унікальності текстів роботи  
та невикористання матеріалів інших авторів без посилання

Коломієць Марія Вікторівна  
Факультет інформаційних та прикладних технологій  
Спеціальність - 122 Комп'ютерні науки  
Освітня програма - Розробка програмного забезпечення та комп'ютерна графіка

ДЕКЛАРАЦІЯ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «Система віддаленого управління робототехнікою на платформі Webots з модульним типом поведінки та SQL базою даних» є написана мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21 – 25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

---

дата

---

підпис здобувача