

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РОЗРОБКА ВІРТУАЛЬНОГО
СЕРЕДОВИЩА ДЛЯ ГІБРИДНОЇ ЛАБОРАТОРІЇ GOLDI
RESEARCH AND DEVELOPMENT OF THE VIRTUAL
ENVIRONMENT FOR GOLDI HYBRID LABORATORY

Виконав: студент 2 курсу, групи КНТ-120м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Яремченко Є.В.

(прізвище та ініціали)

Керівник Пархоменко А.В.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Яремченка Євгенія Віталійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та розробка віртуального середовища для гібридної лабораторії GOLDi. Research and Development of the Virtual Environment for GOLDi Hybrid Laboratory

керівник проєкту (роботи) Пархоменко Анжеліка Володимирівна, к.т.н., доцент,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “05” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи) 13 грудня 2021 року

3. Вихідні дані до проєкту (роботи) завдання, рекомендована література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної галузі. 2. Дослідження технологій реалізації віртуальних середовищ. 3. Проєктування онлайн віртуального середовища Розумний будинок для гібридної лабораторії GOLDi. 4. Конструювання онлайн віртуального середовища Розумний будинок для гібридної лабораторії GOLDi. 5. Методика використання розробленого онлайн віртуального середовища Розумний будинок.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Пархоменко А.В., доцент		
Нормоконтролер	Дейнега Л.Ю., ст.викладач		

7. Дата видачі завдання “ 6 ” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання
2	Аналіз предметної галузі	2 тиждень	Розділ 1
3	Дослідження технологій реалізації віртуальних середовищ	3-4 тижні	Розділ 2
4	Проектування системи	5-6 тижні	Розділ 3
5	Створення програмного коду системи	7-8 тижні	Розділ 4, Додаток А
6	Тестування та відлагодження програмного продукту	9 тиждень	Розділ 4
7	Розробка методики використання програмного продукту	10 тиждень	Розділ 5
8	Оформлення пояснювальної записки	11 тиждень	ПЗ
9	Розробка слайдів презентації	12 тиждень	Додаток Б
10	Нормоконтроль та рецензування	13-14 тижні	
11	Захист роботи	15 тиждень	

Студент

(підпис) Яремченко Є.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Пархоменко А.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
189 с., 13 табл., 54 рис., 2 дод., 113 джерел.

РОЗУМНИЙ БУДИНОК, ГІБРИДНА ЛАБОРАТОРІЯ, ВІРТУАЛЬНЕ
СЕРЕДОВИЩЕ, МЕТОД ВЗАЄМОДІЇ, ІНТЕРФЕЙС КОРИСТУВАЧА,
СЕРВЕР, MQTT, UNREAL ENGINE 4.

Об'єкт дослідження – проведення віддалених та віртуальних експериментів для вивчення систем Розумний будинок та технологій Інтернету речей.

Предмет дослідження – методи та засоби розробки віртуальних середовищ.

Мета роботи – дослідження та програмна реалізація віртуального онлайн-середовища Розумний будинок для розширення функціональних можливостей гібридної лабораторії GOLDi.

Матеріали, методи та технічні засоби: функціональне та об'єктно-орієнтоване програмування; гібридна лабораторія GOLDi; мови програмування C++, Blueprint, JavaScript / TypeScript; ігровий рушій Unreal Engine 4, платформа Node.js для серверної розробки та її фреймворк Express.js; ручне тестування; персональний комп'ютер з процесором AMD Ryzen 5 4600H під управлінням операційної системи Microsoft Windows 10.

Результати. Розроблено програмну систему, яка складається з серверу та онлайн віртуального середовища Розумний будинок, для розширення функціональних можливостей гібридної лабораторії GOLDi.

Висновки. Створене віртуальне середовище надає можливість проведення онлайн-експериментів з вивчення систем Розумний будинок та технологій Інтернету речей.

Галузь використання – навчальний процес підготовки фахівців з інформаційних технологій.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 189 pages, 54 figures, 13 tables, 2 appendixes, 113 sources.

SMART HOUSE, HYBRID LABORATORY, VIRTUAL ENVIRONMENT, METHOD OF INTERACTION, USER INTERFACE, SERVER, MQTT, UNREAL ENGINE 4.

Object of study is conducting remote and virtual experiments to study Smart House systems and the Internet of Things.

Subject of study are methods and tools for developing virtual environments.

The purpose of the work is to research and to develop virtual online environment Smart House in order to expand the functionality of the GOLDi hybrid laboratory.

Materials, methods, and tools: functional and object-oriented programming; GOLDi hybrid laboratory; C++, Blueprint, JavaScript / TypeScript programming languages; game engine Unreal Engine 4, Node.js platform for server development and its Express.js framework; manual testing; a personal computer with an AMD Ryzen 5 4600H processor running the Microsoft Windows 10 operating system.

Results. Developed a software system consisting of a server and an online virtual environment Smart House to expand the functionality of the GOLDi hybrid laboratory.

Conclusions. The created virtual environment provides an opportunity to conduct online experiments to study the Smart House system and the Internet of Things technologies.

The field of application is the educational process of training specialists in information technologies.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	11
1 Аналіз предметної галузі.....	13
1.1 Огляд існуючих технологій онлайн-інженерії.....	13
1.2 Дослідження існуючих віртуальних лабораторій та середовищ для навчання в галузі IoT та систем Розумний будинок.....	21
1.2.1 VL IoTIFY	21
1.2.2 BC Home I/O	23
1.2.3 BC Virtual IOT	26
1.2.4 BC OpenSHS	28
1.2.5 VL Wokwi.....	30
1.2.6 BC UbiSim.....	31
1.2.7 Віртуальний тур для «розумного» будинку	32
1.2.8 BC Розумний будинок	34
1.2.9 Порівняння досліджених віртуальних лабораторій та середовищ для навчання в галузі IoT та систем Розумний будинок.....	36
1.3 Аналіз структурних та функціональних особливостей гібридної лабораторії GOLDi	41
1.4 Постановка задач магістерської роботи.....	53
2 Дослідження технологій реалізації віртуальних середовищ	55
2.1 Дослідження особливостей існуючих віртуальних середовищ	55
2.2 Огляд методів організації взаємодії користувачів з віртуальним середовищем	58
2.2.1 Поняття «людино-машинна взаємодія».....	58
2.2.2 Парадигма WIMP	60
2.2.3 Методи взаємодії, специфічні для віртуальних середовищ	63
2.2.4 Дотикальний інтерфейс користувача.....	65

2.2.5 Узагальнення методів організації взаємодії користувачів з віртуальним середовищем.....	66
2.3 Розробка методу організації взаємодії користувачів з віртуальним середовищем	69
2.4 Аналіз технологій реалізації віртуальних середовищ.....	72
2.4.1 MATLAB.....	72
2.4.2 Unity.....	74
2.4.3 Unreal Engine 4	77
2.4.4 HTML5, WebGL, JavaScript.....	79
2.4.5 Порівняння технологій реалізації віртуальних середовищ	82
2.5 Порівняння протоколів передачі даних, що використовуються в системах Розумний будинок та IoT.....	85
2.5.1 Класифікація протоколів передачі даних	85
2.5.2 CoAP	86
2.5.3 MQTT	87
2.5.4 XMPP.....	88
2.5.5 AMQP	89
2.5.6 Порівняння протоколів передачі даних прикладного рівня, що використовуються в системах Розумний будинок та IoT	90
2.6 Висновки до розділу 2	91
3 Проєктування онлайн віртуального середовища Розумний будинок для гібридної лабораторії GOLDi.....	93
3.1 Аналіз вимог до віртуального середовища	93
3.2 Розробка архітектури програмної системи.....	98
3.3 Розробка архітектури прикладного програмного інтерфейсу.....	104
3.3.1 Точки входу серверу	104
3.3.2 Структура MQTT-тем обладнання віртуального середовища	105
3.4 Обґрунтування вибору інструментарію розробки.....	107
3.4.1 Вибір стеку технологій розробки	107
3.4.2 Вибір інтегрованого середовища розробки.....	110

3.5 Висновки до розділу 3	113
4 Конструювання онлайн віртуального середовища Розумний будинок для гібридної лабораторії GOLDi.....	114
4.1 Розробка алгоритму функціонування системи	114
4.2 Опис особливостей програмної реалізації.....	117
4.2.1 Особливості програмної реалізації серверу	117
4.2.2 Особливості програмної реалізації віртуального середовища.....	121
4.3 Опис структури даних	131
4.4 Розробка графічного інтерфейсу користувача.....	134
4.5 Тестування розробленої системи.....	143
4.6 Висновки до розділу 4	145
5 Методика використання розробленого онлайн віртуального середовища Розумний будинок.....	146
5.1 Умови виконання програмного продукту	146
5.2 Налаштування та запуск програмного продукту	146
5.3 Робота з програмною системою	147
5.4 Висновки до розділу 5	151
Висновки	153
Перелік джерел посилання	155
Додаток А Текст програми.....	167
Додаток Б Слайди презентації	181

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
BPU	– Bus Protection Unit;
ECP	– Experiment Control Panel;
GE	– Game Engine;
GUI	– Graphical User Interface;
HUD	– Heads-Up Display;
IDE	– Integrated Development Environment;
IoT	– Internet of Things;
IP	– Internet Protocol;
MQTT	– Message Queue Telemetry Transport;
PSPU	– Physical System Protection Unit;
REST	– Representational State Transfer;
RL	– Remote Laboratory;
TCP/IP	– Transmission Control Protocol / Internet Protocol;
UE4	– Unreal Engine 4;
UI	– User Interface;
VL	– Virtual Laboratory;
VR	– Virtual Reality;
БД	– база даних;
БК	– блок керування;
ВП	– віртуальний прилад;
ВС	– віртуальне середовище;
ГЛ	– гібридна лабораторія;
ДІК	– дотикальний інтерфейс користувача;
ІС	– ідентифікатор сеансу;
ЛМВ	– людино-машинна взаємодія;
МП	– мова програмування;
ОЛ	– онлайн-лабораторія;

- ОС – операційна система;
- ОТ – онлайн-технології;
- ПАК – програмно-апаратний клієнт;
- ПЗ – програмне забезпечення;
- ПК – персональний комп'ютер;
- ПП – програмний продукт.

ВСТУП

Сьогодні необхідність впровадження віртуальних лабораторій (Virtual Laboratory, VL) в навчальний процес є актуальною темою, інтерес до якої значно підвищився через пандемію COVID-19. VL, які можуть як доповнювати фізично проведений експеримент, так і повністю його відтворювати, можуть бути ефективним та надійним інструментом для навчання в різних галузях освіти [1]-[3].

Однією з таких галузей є навчання системам та технологіям домашньої автоматизації, Розумний будинок та Інтернету речей (Internet of Things, IoT). Постійно зростаючий попит на спеціалістів з даних технологій потребує розробки нових підходів та інструментів для їх навчання. Такий попит, що викликаний стрімким розвитком концепцій «розумних» міст та «розумних» будинків, змушує навчальні заклади відкривати відповідні освітні програми. Оскільки невід'ємним в процесі вивчення даних систем є використання різних пристроїв, сенсорів та мікропроцесорів, створення навчальних VL та віртуальних середовищ (BC) дозволяє значно знизити витрати навчальних закладів на підтримку таких освітніх програм та курсів [4]-[6].

Прикладом навчального закладу, що веде підготовку спеціалістів з систем автоматизації та технологій IoT є Технічний університет Ільменау, Німеччина. У даному університеті у співпраці з університетами з України, Грузії, Вірменії та ін. було реалізовано гібридну лабораторію (ГЛ) GOLDi, що уявляє собою набір віртуальних експериментів та експериментів з використанням віддаленого обладнання. Серед представлених експериментів є експерименти з моделями контролю елеваторів, виробничих ділянок, складів зберігання тощо, але відсутня реалізація комплексного експерименту чи середовища з дослідження систем домашньої автоматизації. Таке середовище могло б бути використано в навчальному процесі викладачами для демонстрації принципів автоматизації систем Розумний будинок за допомогою сценаріїв, а також для здобуття студентами практичних навичок з

розробки апаратного та програмного (ПЗ) забезпечення для роботи з системами Розумний будинок.

Тому тема роботи є актуальною.

Мета роботи – дослідження та програмна реалізація віртуального онлайн-середовища Розумний будинок для розширення функціональних можливостей гібридної лабораторії GOLDi.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Огляд існуючих технологій онлайн-інженерії

Сьогодні онлайн-технології (ОТ) є ядром більшості галузей інженерії та всього суспільства і нерозривно пов'язані, наприклад, з IoT, Індустрією 4.0, Індустріальним Інтернетом речей, хмарними технологіями, науками про дані, змішаною реальністю, дистанційними робочими середовищами, біомедичною інженерією тощо. Онлайн-технології (Інтернет-технології) – це технології, що використовують одну або декілька комп'ютерних мереж, які використовують TCP/IP, IP або іншу архітектуру, або які використовують комп'ютерні термінали, термінальні сервери, маршрутизатори, технологію багатоадресної передачі або інший пристрій обробки або передачі даних через кабель, провід, бездротовий зв'язок, телевізійні приставки або інші технології, незалежно від пристрою перегляду кінцевого користувача [7]-[8].

Важливою галуззю використання ОТ є онлайн-інженерія. В цілому онлайн-інженерія – це майбутнє техніки і науки. Використання кіберфізичних систем, віртуальних приладів (ВП) і пристроїв з дистанційним керуванням, а також віддалених лабораторій (Remote Laboratory, RL) – це напрямки для передових середовищ віддаленої роботи. Через зростаючу складність інженерних задач, появу дедалі більшої кількості спеціалізованого і дорогого устаткування, а також програмних інструментів і симуляторів, брак висококваліфікованого персоналу, а також потребу глобалізації та спільної роботи стає важливим використання хмарних технологій для максимального збільшення продуктивності використання інженерних ресурсів. Онлайн-інженерія – це спосіб вирішити ці проблеми [9].

Отже, онлайн-інженерія – це сучасна тенденція в інженерії і науці, що сфокусована на розробці методологій, технік та інструментів, метою яких є дозволити і організувати спільне використання інженерного обладнання та ресурсів, а також спеціалізованого ПЗ (наприклад, симуляторів), за допомогою ОТ [9].

Онлайн-інженерія – це узагальнене визначення для широкого кола суміжних технологій, а саме: віддалена інженерія, онлайн-лабораторії (ОЛ), віртуальні прилади та онлайн-моделювання, хмарні технології, хмарне проєктування і виробництво, колаборативні мережі тощо [9].

Віддалена інженерія – це технології, які дозволяють виконувати технічні операції з використанням обладнання частково або повністю з віддаленої відносно обладнання локації за допомогою ОТ. Віддалена інженерія може бути хорошим рішенням для технологічних компаній з декількома територіально рознесеними об'єктами зі значною економією коштів і часу з точки зору спільного використання спеціалізованого обладнання та обслуговуючого персоналу. Такі технології також можуть дозволити невеликим, навіть сімейним компаніям, отримати доступ до віддаленого інженерного центру [10]-[11].

Онлайн-лабораторія – це апаратно-програмна система, яка дозволяє проводити експерименти без фізичного контакту з реальним обладнанням або його симуляцією за допомогою ОТ. ОЛ можуть бути представлені у вигляді віддалених, віртуальних чи гібридних лабораторій [12].

Одним з популярних напрямків використання засобів віддаленої інженерії є віддалені лабораторії. Віддалена лабораторія – це онлайн-лабораторія, яка надає віддалений доступ до фізичної лабораторії. Безпечний і надійний доступ забезпечується вебсистемою для дистанційного керування спеціалізованим обладнанням. RL дозволяють дистанційно проводити експерименти з реальними вимірювальними приладами і реальним об'єктом дослідження, надаючи результати, точність котрих відповідає результатам, отриманим у традиційних фізичних лабораторіях. Такі лабораторії використовуються тоді, коли експеримент або інструмент фізично недоступні через розташування або небезпеку. Це також надає можливість поділитися дорогим обладнанням та іншими інженерними ресурсами з іншими дослідниками [13]-[15].

RL виступає новим рівнем у процесі взаємодії користувача з обладнанням фізичної лабораторії та її ресурсами. Вона відповідає за передачу дій користувача та отримання сенсорної інформації від обладнання і зазвичай реалізується на базі клієнт-серверної архітектури [16]-[17].

З боку користувача віддалена інфраструктура виконує наступні ключові функції:

- надає інтерфейс користувача (User Interface, UI), який дозволяє контролювати і керувати експериментальним обладнанням;
- керує доступом до лабораторії, щоб гарантувати, що одночасно тільки один користувач може проводити експеримент [16].

З боку обладнання віддалена інфраструктура виконує наступні ключові функції:

- веде моніторинг пристроїв, наприклад, за допомогою відеокамер, мікрофонів та інших датчиків;
- керує пристроями, наприклад, за допомогою інтерфейсів вводу / виводу, двигунів або інших виконавчих механізмів;
- гарантує, що експеримент буде «очищений» в кінці сеансу користувача шляхом автоматичного перезавантаження пристроїв або переведення їх у стабільний стан [16].

В останні роки RL набули широкого поширення, особливо в сфері інженерної освіти. В основному це пов'язано зі значними перевагами, які вони надають в порівнянні з їх традиційними аналогами, а саме:

- покращений доступ користувачів і пов'язане з цим зростання використання;
- можливість розподілу ресурсів між установами для компенсації витрат;
- наявність більш різноманітного спектра експериментів;
- зменшення ризиків з питань безпеки під час використання;
- керований доступ, який обмежує навмисне або ненавмисне неправомірне використання [18].

Однак дослідження, описані в роботі [6], показали, що через віддалене використання реального обладнання тільки один користувач може проводити експерименти на певному віддаленому пристрою. В результаті можуть виникати черги користувачів. Крім цього, важливий недолік RL виявляється в разі використання їх в освітньому процесі, оскільки відповідно до теорії навчання конструктивізму студенти повинні перебувати в реальних умовах [18].

Іншим підходом до реалізації ОЛ є віртуальні лабораторії. Віртуальна лабораторія – це розподілене середовище моделювання, яке призначене для інтерактивного моделювання математичної моделі реальної системи, тобто це віддалено доступна імітаційна лабораторія. VL можна використовувати для експериментів, для яких зазвичай потрібне занадто дороге, небезпечне або недоступне обладнання (наприклад, ядерний реактор). Крім цього, такі лабораторії можна використовувати для демонстрації явищ, які складно виразно продемонструвати у традиційній чи віддаленій лабораторії. VL також дозволяють повторювати експеримент кілька разів, даючи можливість побачити, як змінені параметри і налаштування впливають на результат. Одна з важливих функцій VL – дати можливість отримати результати, максимально наближені до реальних, не завдаючи при цьому реального збитку в разі помилки в процесі проведення експерименту [12], [19].

Одна з основних переваг використання VL полягає в тому, що їх можна легко відтворити. Одночасно може бути запущено багато копій експерименту, що вирішує проблему виникнення черг для віддаленого доступу до обладнання під час використання RL. Крім цього, модельований експеримент у VL можливо поставити на паузу, що збільшує можливості користувачів для проведення повноцінного та різнопланового аналізу об'єкту дослідження [19].

Як вже було зазначено раніше, проблеми безпеки, пов'язані з обмеженнями змінних процесу моделювання експерименту, не так важливі, як в реальних експериментах, оскільки це не може призвести до травмування. Тому ще одна перевага використання VL полягає в тому, що є можливість

прискорити роботу систем з повільною динамікою для швидкої візуалізації, хоча в реальності такі б зміни могли б призвести до негативних наслідків [19].

Одним з основних недоліків VL є те, що вони не можуть повністю відтворити реалістичну поведінку об'єкту дослідження, оскільки часто використовують ідеалізовані моделі, які не в повному обсязі відповідають реальним ситуаціям, явищам та моделям. Ще одним недоліком VL (так само як і RL) є те, що через технічні обмеження вони зазвичай не надають необхідної гнучкості в проведенні експериментів, оскільки їх найчастіше створюють спеціально для проведення обмеженого набору експериментів [19]-[20].

У літературі часто для позначення систем типу Віртуальна лабораторія використовується термін «віртуальне середовище», хоча це різні поняття. Віртуальне середовище – це віртуальний онлайн-світ (хоча, так само як і VL, може бути офлайн), що складається з інтерактивних графічних 3D-моделей. Це середовище супроводжується методами імерсивної візуалізації та інструментами для миттєвого маніпулювання в цьому віртуальному світі. Дії ігрового персонажу у ВС знаходяться під контролем користувача в реальному часі [21]-[22].

Створення реалістичних ВС – складний, дорогий і трудомісткий процес. Хоча для розробки ВС доступні різні технології, багато з них надають лише частину необхідних інструментів, необхідних для створення повноцінних віртуальних світів [23].

Крім створення моделей віртуальних об'єктів, наприклад, будівель і декорацій, розробник також повинен розробити механізми взаємодії користувача з середовищем, необхідну логіку роботи та поведінки середовища і об'єктів (наприклад, виявлення зіткнень), і всі інші невізуальні функції віртуального світу, наприклад, звукові сигнали [23].

Деякі особливості ВС, такі, як вітер, вогонь, дим і воду, а також поведінку інших неігрових персонажів, особливо важко моделювати. Крім

того, інструменти для створення ВС вимагають значних навичок програмування і великих витрат часу з боку розробника [23].

В основі VL та ВС знаходиться технологія моделювання. Моделювання в реальному часі – це комп’ютерна модель, яка відображає певну математичну модель в реальному часі, тобто час роботи симуляції відповідає часу, який займає процес чи явище, для якого відбувається симуляція. Моделювання також може відбуватися на основі реальних даних. В такому випадку мова йде про онлайн-моделювання (онлайн-симуляцію). Онлайн-симулятор – це категорія імітаційних програм, в яких імітаційна (математична) модель пов’язана онлайн з реальним об’єктом чи процесом моделювання та які надають різні засоби візуалізації моделі [24]-[25].

Одним з фундаментальних завдань онлайн-моделювання є оперативне постачання даних з реальної системи в імітаційну модель. На початку моделювання імітаційна модель повинна відтворювати актуальний стан реальної системи, тобто дані в імітаційній моделі повинні бути співставленні з даними про стан об’єктів з реальної системи. Саме тому типовою сферою застосування онлайн-моделювання є системи підтримки прийняття рішень для задач планування в виробничих системах (планування в реальному часі) [24].

Звичайне моделювання ґрунтується на припущеннях про лінійність і стаціонарність реальної системи. На практиці багато систем демонструють нелінійну і нестаціонарну поведінку, тому в онлайн-симуляції часто потрібно використовувати адаптивну нелінійну модель. Онлайн-моделювання постійно коригує імітаційну модель на основі вхідних даних, завдяки чому зміни у поведінці нестаціонарної системи фіксуються симуляцією. Онлайн-моделювання нестаціонарних і нелінійних систем зазвичай являє собою складну задачу. Поширений підхід полягає у використанні адаптивних алгоритмів для відстеження тимчасових змін системи [26].

Найефективнішим підходом до реалізації ОЛ як для досліджень, так і для навчання є гібридні лабораторії. Гібридна лабораторія – це онлайн-лабораторія, яка поєднує технології RL та VL. ГЛ намагаються

використовувати деякі переваги RL та VL для того, щоб забезпечити більшу реалістичність, економічність та інтерактивність (за рахунок використання ігрових механік комп'ютерних ігор). Крім цього, поєднання даних технологій реалізації ОЛ дозволяє забезпечити деякі функції ВС [27]-[28].

Створення ГЛ, особливо кросплатформних, в даний час являє собою серйозну проблему. Хоча RL та VL самі по собі є відомими та добре дослідженими у літературі технологіями, ГЛ менш ретельно досліджені. Це частково пов'язано з тим, що до недавнього часу графічні технології в браузерях були обмежені і часто покладалися на сторонні технології та плагіни. Сьогодні в літературі представлено малу кількість добре досліджених та розроблених моделей, архітектур або керівництв для створення нових ГЛ. Крім цього, деякі з існуючих ГЛ часто представлені як системи для доказу концепції, а не як завершене та готове для використання ПЗ, або не підтримуються розробниками і є закритими для громадськості [27].

Крім технологій, які використовуються у RL та VL, ГЛ можуть використовувати технологію віртуальних приладів. Віртуальні прилади – це міждисциплінарна галузь, в якій об'єднані сенсорні, апаратні і програмні технології з метою створення гнучких і складних інструментів для програм керування і моніторингу. Ця нова концепція заснована на ідеї створення нового власного інструменту для користувача, який замінює традиційну адаптацію потреб користувача до існуючих інструментів. Як правило, ВП складається зі стандартного для галузі комп'ютера або робочої станції, оснащеного потужним прикладним ПЗ, рентабельним обладнанням і програмним драйвером, які разом виконують певні функції традиційного приладу. ВП являють собою фундаментальний перехід від традиційних апаратних вимірювальних систем до програмно-орієнтованих систем, які використовують обчислювальну потужність, продуктивність, графічні можливості й інші види зв'язку ПК і робочих станцій [29]-[31].

Найважливішим компонентом ВП є ПЗ з призначенням для користувача ергономічним інтерфейсом. Галузі, в яких можна використовувати ВП,

охоплюють практично всі види діяльності людини. Ось чому ПЗ, яке розробляється для надання можливості проєктування та розробки конкретних користувальницьких систем на основі ВП, має пропонувати велику кількість опцій і інструментів, в тому числі деякі, які можуть виглядати несумісними в конкретній системі. В ідеальному режимі таке ПЗ повинне організувати існуючі програмні інструменти у такий спосіб, щоб розробник міг легко вибирати конкретні параметри і керувати ними при виборі параметрів конфігурації розроблюваної системи. За допомогою відповідного програмного інструменту користувачі можуть ефективно створювати свої власні програми, проєктуючи і інтегруючи процедури, необхідні для конкретного процесу. Вони також можуть створити відповідний графічний інтерфейс користувача (Graphical User Interface, GUI), який найкращим чином відповідає цілям програми та тих, хто буде з нею взаємодіяти [30].

Отже, на основі проведених досліджень запропоновано класифікацію технологій онлайн-інженерії, яку представлено на рисунку 1.1. Крім цього, було зроблено висновок, що ВС та VL можуть бути використані для розробки віртуальних систем для навчання студентів в галузі IoT та систем Розумний будинок.



Рисунок 1.1 – Класифікація технологій онлайн-інженерії

1.2 Дослідження існуючих віртуальних лабораторій та середовищ для навчання в галузі IoT та систем Розумний будинок

1.2.1 VL IoTIFY

IoTIFY – це перша в галузі хмарна платформа для тестування продуктивності, призначена для допомоги в створенні, перевірці та безперервному моніторингу сучасних корпоративних систем IoT [32].

Традиційне навантажувальне тестування для вебзастосунків та прикладного програмного інтерфейсу (Application Programming Interface, API) абсолютно не підходять для тестування рішень IoT. IoTIFY розроблений з нуля для задоволення конкретних вимог до тестування та моделювання великомасштабних систем IoT – для навантажувального, функціонального тестування, тестування продуктивності та безпеки систем IoT [32].

Дана платформа дозволяє створювати реалістичні моделі пристроїв з батареєю, датчиками та різними параметрами, які можливо налаштувати. Завдяки цьому можливо моделювати взаємодію пристроїв з навколишнім середовищем і керувати цим процесом динамічно через API [32].

IoTIFY підтримує найпопулярніші протоколи передачі даних, такі, як MQTT, HTTP, CoAP, AMQP та LWM2M. Завдяки підтримці Raw UDP / DTLS і TCP / TLS також можливо моделювати будь-який інший протокол [32].

Крім цього, IoTIFY вимірює найбільш важливі метрики для тесту і дозволяє вимірювати і візуалізувати власні метрики тесту за допомогою простого API [32].

IoTIFY пропонує два рішення – VL та симулятор IoT. VL IoTIFY – це онлайн-платформа для університетів для навчання технологіям IoT. Це повністю інтерактивний робочий онлайн-простір для створення електронних прототипів за допомогою веббраузера. Дана лабораторія дозволяє розробляти прошивку і застосунки IoT в браузері на поширених вбудованих апаратних платформах. Використовуючи дану VL, студенти можуть розробляти

прототипи пристроїв IoT, вивчати програмування вбудованих систем і створювати свої власні прототипи [33].

VL IoTIFY також забезпечує можливість спільної роботи над проектом. Оскільки всі компоненти доступні онлайн через браузер, стає легко працювати у команді над одним проектом. Для викладачів це означає кращий досвід викладання. Для учасників проекту це означає більш приємний робочий процес створення прототипів [34].

VL IoTIFY підтримує мікроконтролерні платформи Arduino Uno, STM32 Nucleo F411RE і Raspberry Pi, а також ряд периферійних приладів [34].

VL IoTIFY доступна за університетською ліцензією мінімум на 10 студентських ліцензій на рік. Для індивідуального використання спочатку доступно 2 кредити (1 кредит – 1 проєкт на 1 годину використання), потім потрібно купляти кредити додатково [33].

На рисунках 1.2 та 1.3 зображено інтерфейс VL IoTIFY [34].

Симулятор IoT від IoTIFY вирішує проблему тестування масштабованості та продуктивності системи IoT, використовуючи віртуалізацію та моделюючи тисячі кінцевих точок IoT у хмарі, тим самим надаючи можливість повного моделювання системи до того, як продукт потрапить до замовника. Проте це рішення не є VL чи BC [35].

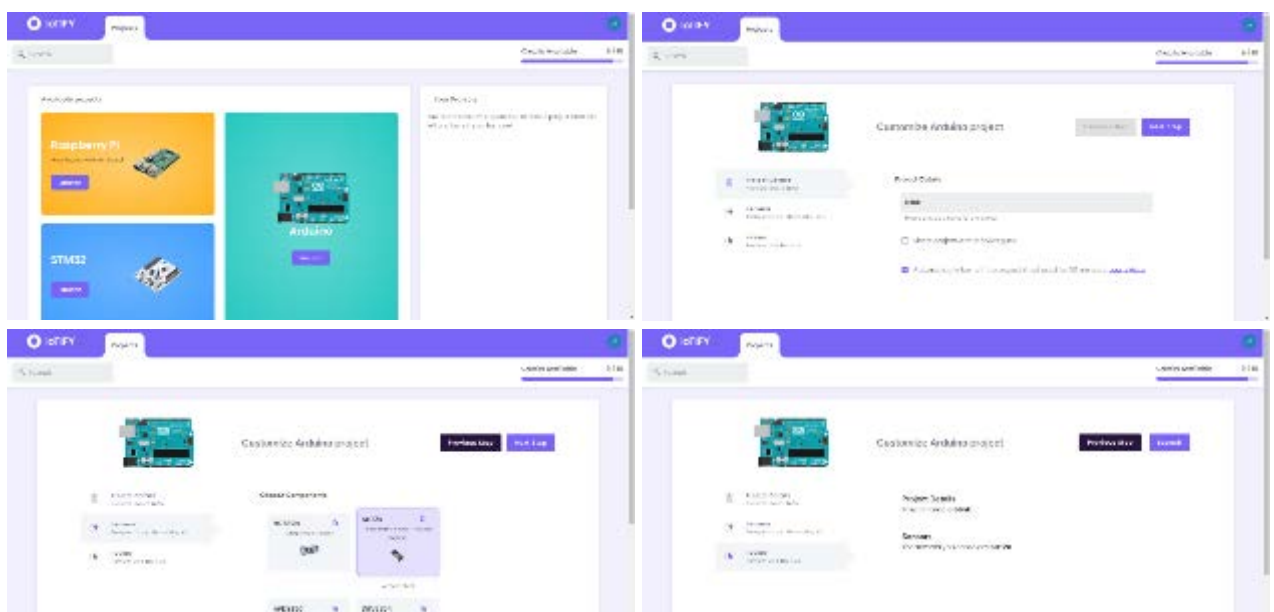


Рисунок 1.2 – Інтерфейс створення нового проєкту у VL IoTIFY [34]



Рисунок 1.3 – Інтерфейс робочого простору VL IoTIFY [34]

Отже, IoTIFY це платформа, яка надає хмарний сервіс для тестування розроблених IoT-систем, а також VL, за допомогою якої можна покращити навички в розробці IoT-пристроїв [32].

1.2.2 BC Home I/O

Home I/O – це віртуальне середовище Розумний будинок та його довкілля. Воно розроблене для охоплення широкого кола навчальних цілей в галузі природничих наук, технологій, інженерії тощо. Home I/O має все необхідне для моделювання та моніторингу «розумного» будинку в реальному часі. На рисунках 1.4 та 1.5 зображено інтерфейс BC Home I/O [36]-[37].

Основна мета Home I/O – познайомити її користувачів з концепціями домашньої автоматизації за допомогою інтерактивного «розумного» будинку. Home I/O оснащений найпоширенішими пристроями автоматизації. Завдяки цьому дане ВС надає студентам можливість розробити рішення для керування системою домашньої автоматизації і зрозуміти вплив їх впровадження [37].

Home I/O пропонує ліцензії для приватних осіб і організацій. Індивідуальні ліцензії доступні тільки для особистого використання. Корпоративна ліцензія повинна бути придбана для кожного окремого

навчального закладу або компанії. Також можливо отримати безкоштовний пробний термін 30 днів [36].



Рисунок 1.4 – Інтерфейс ВС Home I/O [37]



Рисунок 1.5 – Елементи керування ВС Home I/O [37]

Home I/O надає наступні можливості:

- створення сценаріїв системи Розумний будинок за допомогою поширених пристроїв автоматизації. Наприклад, налаштування розкладу або умов увімкнення / вимкнення світла, коли люди входять в кімнату або виходять з неї; автоматична перевірка стану дверей та вікон (відкрито / закрито);
- зміна налаштувань погоди, дати, часу і місця. Це надає можливість побачити, як ці параметри впливають на ВС, і проаналізувати результати. Моделювання теплового балансу в реальному часі враховує погодні умови, місце розташування та властивості будинку;
- можливість прискорювати час і змодельовати день за 17 секунд;
- 174 пристроїв. Home I/O дозволяє взаємодіяти з освітлювальними, опалювальними та іншими пристроями системи Розумний будинок, використовуючи понад 400 точок введення / виведення;
- керування світлом за допомогою цифрових значень або шляхом обчислювання яскравості на основі аналогових значень;
- підсистема Power Panel, яка відстежує дані про споживання енергії в режимі реального часу і враховує погодинне, денне, щотижнєве і щомісячне споживання енергії і її вартість;
- підсистема Connect I/O дозволяє інтегрувати Home I/O з широким спектром зовнішніх технологій автоматизації (наприклад, PLC, Modbus, мікроконтролери та багато інших);
- програмування за допомогою Python, Scratch 2 та Scratch 3 для навчання принципам програмованої взаємодії з системами та приладами «розумного» будинку [36].

Отже, ВС Home I/O – це потужне ПЗ для проведення досліджень та навчання в галузі систем IoT та Розумний будинок [36].

1.2.3 BC Virtual IOT

Virtual IOT – це віртуальне середовище для колаборативного навчання технологіям IoT. Мета проєкту – розробити віртуальне тривимірне лабораторне середовище і забезпечити навчання технологіям IoT на цій платформі [38].

У результаті цього проєкту планується створити віртуальний 3D-клас, в якому студенти будуть вивчати технології IoT. Крім цього, будуть представлені інноваційні та різні навчальні матеріали, які можна буде використовувати під час занять. Крім віртуального класу, буде створений віртуальний центр, де студенти будуть займатися дослідженнями і розробками [38].

Цей проєкт здійснюється на міжнародному рівні, оскільки обмін передовим досвідом у навчанні технологіям IoT є однією з основних цілей. Навчальні заклади, що беруть участь у розробці Virtual IOT, вже ведуть підготовку спеціалістів з даного напрямку. Але проведення міжнародних спостережень, вивчення різних практик викладання з різних країн може дати цінні результати для учасників. Завдяки діяльності в межах проєкту викладачі з кожної установи-партнера отримають можливість взяти участь у міжнародному заході з навчання і побачити різні платформи, які використовуються для навчання IoT в різних країнах [38].

Для того, щоб почати використання BC Virtual IOT потрібно зареєструватися на порталі проєкту в ролі викладача. Після проходження процесу реєстрації потрібно додати студентів до класу. Студентам потрібно завантажити BC на ПК та авторизуватися, використовуючи ту поштову скриньку, яку викладач використав для запрошення студента [39].

Навчальний курс складається з трьох рівнів з трьома завданнями на кожному з них. Мета студента – пройти рівень, отримавши якнайбільше балів за проходження завдань. Перехід до наступного рівня можливий лише після виконання всіх завдань на поточному рівні. Навчальний матеріал

представлений у вигляді відео, які можна переглянути у кімнаті, що знаходиться перед кімнатою з завданням [39].

На рисунках 1.6 та 1.7 зображено інтерфейс ВС Virtual IOT.

Отже, ВС Virtual IOT – це амбіційний проєкт, але на даний момент ВС знаходиться в процесі розробки. Тому є проблема з відображенням навчальних відеоматеріалів, чатом та з віртуальним оточенням [38].

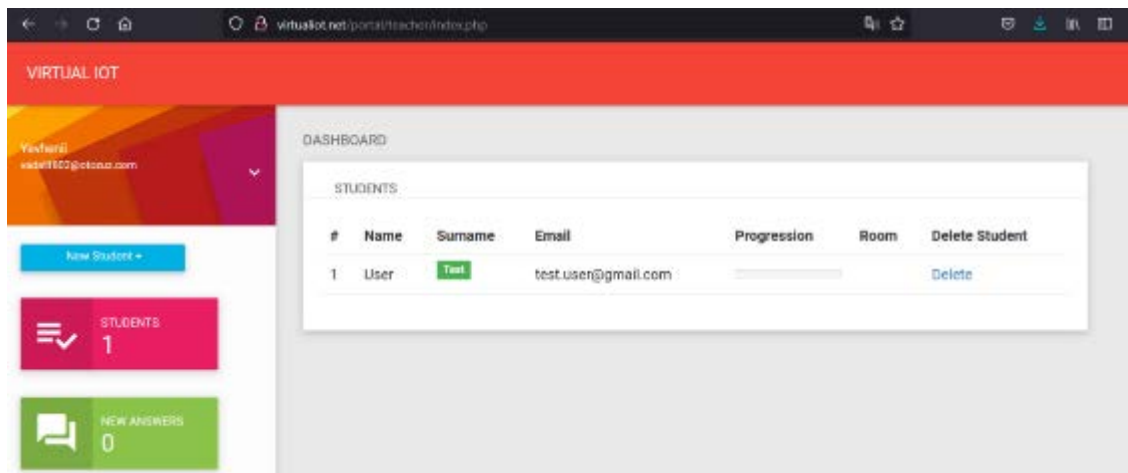


Рисунок 1.6 – Інтерфейс викладача для створення нового навчального класу
ВС Virtual IOT



Рисунок 1.7 – Інтерфейс ВС Virtual IOT

1.2.4 BC OpenSHS

Open Smart Home Simulator (OpenSHS) – це відкритий кросплатформний 3D-симулятор віртуального середовища Розумний будинок для створення наборів даних від пристроїв та датчиків IoT для подальшого аналізу. BC OpenSHS пропонує дослідникам в галузі IoT і машинного навчання можливість протестувати і оцінити свої моделі IoT-систем. OpenSHS поєднує в собі переваги як інтерактивного, так і симулятивного підходу. Такий підхід скорочує час і зусилля, необхідні для створення змодельованих наборів даних для «розумного» будинку. Для даного BC були розроблені алгоритми реплікації для створення великих репрезентативних наборів даних для «розумного» будинку. Набір даних, створений за допомогою OpenSHS, можна розширити, не впливаючи на логічний порядок подій. Крім цього, OpenSHS надає досліднику бібліотеку IoT-пристроїв, яка спрощує моделювання різноманітних за конфігурацією систем Розумний будинок [40].

Процес підготовки BC OpenSHS до проведення експерименту складається з трьох окремих етапів:

- а) проєктування конфігурації BC шляхом створення 3D-моделі будинку у 3D-редакторі Blender, розміщення в ній IoT-пристроїв з бібліотеки і створення контексту;
- б) моделювання контекстно-залежних подій;
- в) агрегування, тобто застосування дослідником алгоритму реплікації для створення фінального набору даних [40].

Складним завданням для подібних систем моделювання «розумного» будинку є підтримка декількох жителів будинку. Це складне завдання як для інструменту моделювання, так і для дослідників. В даний час OpenSHS пропонує часткову підтримку симуляції декількох жителів. У майбутньому планується інтегрувати технології віртуальної реальності (Virtual Reality, VR) у BC OpenSHS для підвищення реалістичності. Також планується перенесення реалізації OpenSHS для роботи у веббраузері [40].

Початковий код ВС OpenSHS доступний за посиланням [41]. На рисунках 1.8 та 1.9 зображено інтерфейс ВС OpenSHS та процес проєктування конфігурації «розумного» будинку [40]-[41].

Отже, ВС OpenSHS – це потужний інструмент для дослідників та розробників систем IoT та Розумний будинок, який дозволяє проводити тестування розроблених проєктів [40].



Рисунок 1.8 – Інтерфейс ВС OpenSHS [40]

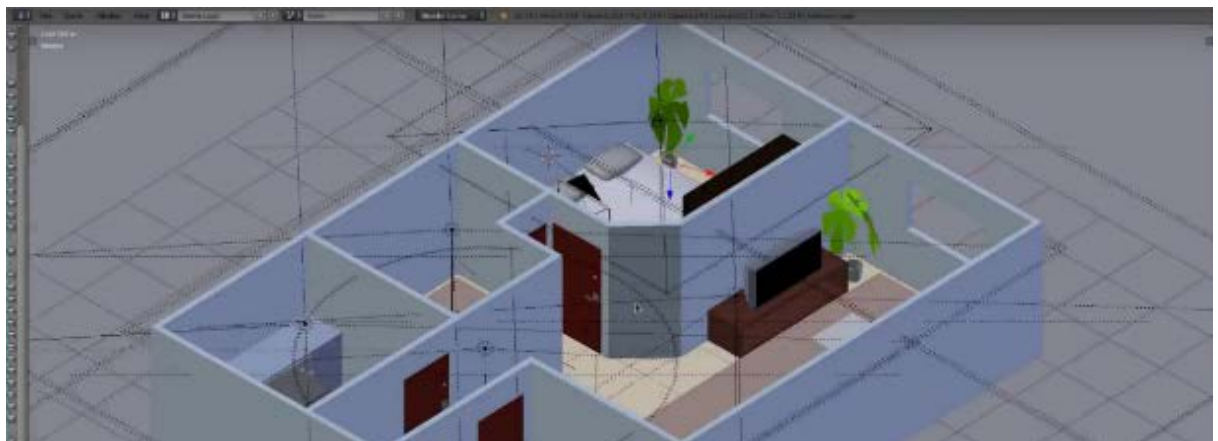


Рисунок 1.9 – Процес проєктування конфігурації ВС OpenSHS у 3D-редакторі Blender [41]

1.2.5 VL Wokwi

Wokwi [42] – це віртуальна онлайн-лабораторія для симуляції Arduino та іншої електроніки. Мета проєкту – надати зручний та потужний інструмент для вивчення програмування на базі платформ Arduino UNO, Mega, Nano та деяких інших, створення прототипів різних пристроїв, зокрема, IoT-пристроїв. Wokwi може моделювати різні прилади, такі, як світлодіоди, регістри зсуву, датчики (ультразвукові, температури, вологості тощо), LCD-дисплей, клавіатуру і багато іншого [43].

Важливою ідеєю проєкту є також дозволити користувачам ділитися своїми проєктами з іншими розробниками, оскільки в результаті обговорення можуть виникнути пропозиції та зауваження, які зможуть допомогти покращити пристрій. Крім цього, VL Wokwi пропонує багато різних готових прикладів реалізації пристроїв, які використовують різні апаратні платформи та вимірювальні прилади [43].

Документація даної VL все ще перебуває на початковій стадії, і в ній відсутньо багато інформації. У разі потреби по допомогу можна звернутися на сервер проєкту у Discord (платформа для спілкування). На рисунках 1.10 і 1.11 зображено інтерфейс VL Wokwi та деякі представлені ресурсом приклади [43].

Отже, VL Wokwi – це корисна платформа для колаборативного навчання, на якій можна поділитися своїми ідеями та побачити ідеї інших [43].

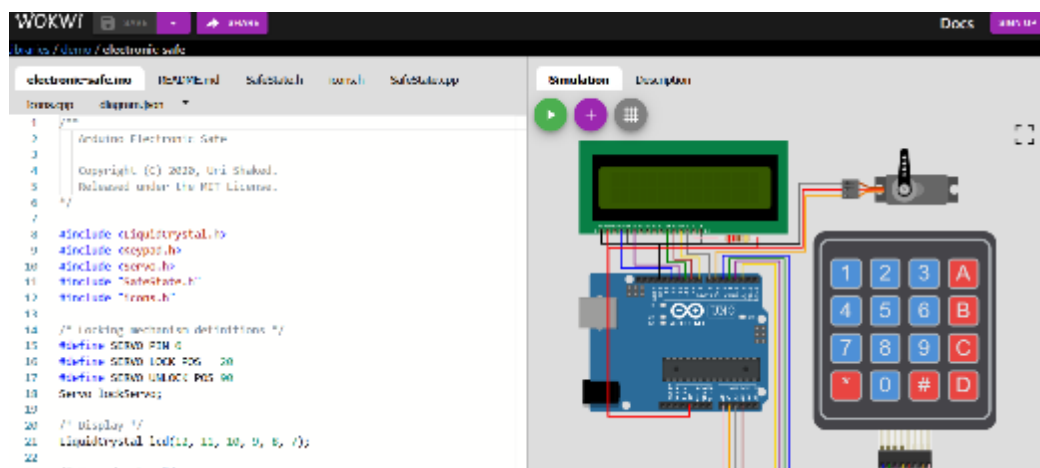


Рисунок 1.10 – Інтерфейс робочого простору VL Wokwi

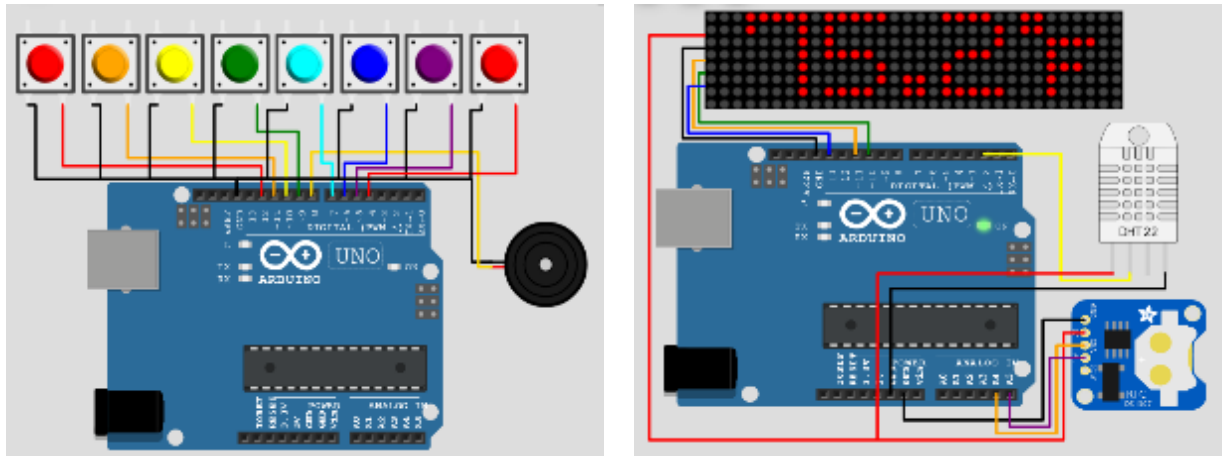


Рисунок 1.11 – Приклади проєктів, створених за допомогою VL Wokwi

1.2.6 BC UbiSim

Дослідникам та розробникам систем Розумний будинок важливо мати великі набори даних із реальних середовищ. Однак не всі дослідники мають достатній бюджет для створення тестових стендів. Цим дослідникам потрібний тренажер, який може синтезувати реалістичні набори даних телеметрії. Для вирішення цієї проблеми в роботі [44] запропоновано симулятор UbiSim для дослідження розпізнавання активності. Дослідження активності можливо проводити за допомогою різних датчиків, включно з датчиками руху, тиску, вібрації, температури і контакту, а також RFID-мітками (мітки для радіочастотної ідентифікації) та приймачами [44].

BC UbiSim дозволяє створити віртуальний «розумний» будинок, у якому можна розмістити незалежні один від одного віртуальні датчики, які імітують отримання та передачу даних так само, як і в справжніх системах. На основі отриманих даних дослідники можуть вивчати взаємодію між агентом (ігровим персонажем BC) та «розумним» будинком. Збір даних виконується завдяки системі датчиків, які розпізнають рухи та дії агента і які користувач встановлює сам на мапі будинку. Весь процес моделювання відстежується в режимі реального часу, щоб переконатися, що конфігурація «розумного» будинку та датчики налаштовані коректно, і візуалізується шляхом зміни розміру та кольору датчиків у разі взаємодії агента з ними (рис. 1.12а) [44].

UbiSim надає тривимірний GUI у вигляді віртуального будинку для інтуїтивно зрозумілого налаштування «розумного» будинку та забезпечення його просторового сприйняття. Усі операції виконуються мишею. На рисунку 1.12б показано інтерфейс налаштування плану будинку та вибору елементів, які будуть розташовані у ньому (вікна, двері тощо). У лівому вікні представлено план створеного будинку зверху. Праве вікно із кнопками забезпечує повний огляд будинку. За допомогою кнопок користувач може додавати елементи на мапу будинку [44].

Отже, ВС UbiSim – це дослідницький інструмент для створення реалістичних наборів даних телеметрії. Недоліком даної системи є те, що в переліку обладнання, доступного для розміщення в будинку, відсутні вимикачі та інші актуатори для керування обладнанням [44].



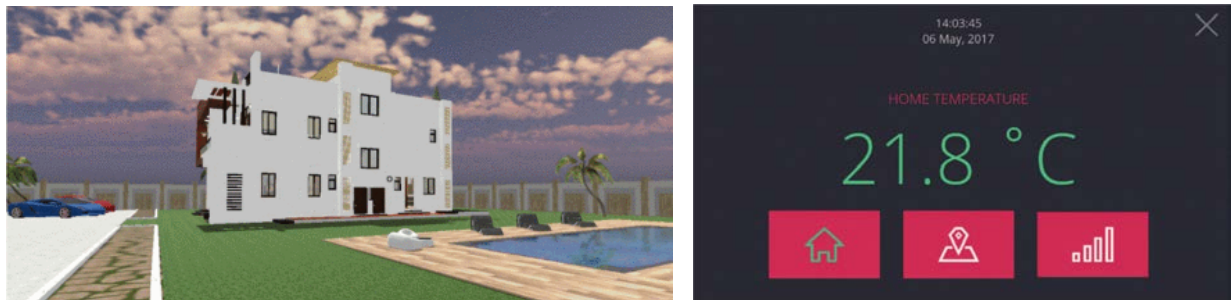
а) процес симуляції; б) налаштування конфігурації системи

Рисунок 1.12 – Інтерфейс ВС UbiSim [44]

1.2.7 Віртуальний тур для «розумного» будинку

У роботі [45] представлено інтерактивний 3D-застосунок, що пропонує віртуальний тур «розумним» будинком та його підвір'ям (рис. 1.13а). Програма реалізована за допомогою Unity. Дане ВС пов'язане через USB-порт з мікроконтролером Arduino, до якого підключено декілька датчиків та

виконавчих пристроїв. Ці електронні пристрої реагують на події у реальному світі та відображають їх у ВС та навпаки [45].



а

б

а) екстер'єр ВС; б) центральна панель керування будинком

Рисунок 1.13 – Інтерфейс віртуального туру для «розумного» будинку [45]

До мікроконтролера Arduino під'єднані датчики пожежі, звуку, температури та освітленості, інфрачервоний приймач, LED-індикатор та п'єзодинамік [45].

Дана система надає наступний функціонал:

- вільне переміщення будинком та його подвір'ям за допомогою стрілок на клавіатурі та комп'ютерної миші. Проте важливо зауважити, що звичайний курсор відсутній. Натомість є точка у центрі екрану, яку за допомогою рухів мишкою можна наводити на об'єкти. Це зроблено для того, щоб у майбутньому було простіше пристосувати дану ВС для роботи з VR-шоломами;
- можливість переглядати журнал показників температури, тобто користувач має доступ до логів системи. Доступ до цих даних реалізований через центральну панель керування будинком (рис. 1.13б), яку можна знайти на кожному поверсі. Для взаємодії з нею потрібно підійти до неї, навести центральну точку за допомогою миші та натиснути ліву кнопку миші;
- можливість налаштування COM-порту для мікроконтролера Arduino;
- можливість перегляду поточної температури;

- використання різних датчиків для вимірювання показників реального середовища та реалізація на основі цих даних різних сценаріїв поведінки ВС;
- можливість керування системою освітлення шляхом плескання у долоні, на які реагує спеціальний датчик звуку, що встановлений на Arduino;
- спрацювання пожежної сигналізації у ВС при виявленні пожежі у реальному світі завдяки датчикам [45].

Отже, віртуальний тур для «розумного» будинку дає уявлення про те, як може відбуватися взаємодія людей з системами Розумний будинок у майбутньому [45].

1.2.8 ВС Розумний будинок

У роботі [46] представлено ВС для навчання технологіям «розумного» будинку, які стосуються збору та обробки різних типів даних та використання різних засобів керування за допомогою хмарних технологій. Представлене ВС Розумний будинок (рис. 1.14) автоматизовано керується за допомогою ВП LabVIEW (ПЗ для розробки ВП). ВС створено на основі ВС Home I/O [36], [46].

Система має наступний функціонал:

- контроль і регулювання температури в будинку;
- вільне переміщення у ВС за допомогою клавіатури та миші;
- автоматичне регулювання яскравості світильників у кожній кімнаті та зовнішнього освітлення;
- керування системою сигналізації з її датчиками руху, датчиками відкриття дверей та вікон для підвищення безпеки та захисту мешканців [46].

Щоб підкреслити можливість використання розробленої технології в реальному середовищі, було реалізовано апаратну частину системи з використанням пристрою MyRIO. Пристрій MyRIO від NI використовується для демонстрації основ електроніки та роботи з датчиками, а також для навчання розробці ВП у середовищі LabVIEW. Цей пристрій може бути

підключений до ПК через USB-кабель або за допомогою бездротової мережі для обробки даних вимірювань у програмі, створеній у LabVIEW [46].



а

б

а) екстер'єр; б) інтер'єр

Рисунок 1.14 – Інтерфейс ВС Розумний будинок [46]

Для демонстрації MyRIO використовується з додатковою платою, DIP-перемикачами, потенціометром, RGB-світлодіодом та фотоелементом для імітації реального існуючого обладнання в будинку. Перемикачі можуть бути призначені для виконання різних завдань у симуляторі, таких, як керування ввімкненням / вимкненням світла, сигналізацією тощо. RGB-світлодіод може використовуватися для індикації стану різних підсистем. Фотоелемент додано для отримання реальних даних освітленості, які можуть бути оброблені в LabVIEW для автоматизації освітлення у ВС [46].

Розроблене ВС призначене для охоплення широкого кола цілей навчальної програми у галузі науки, техніки та інженерії. Завдяки використанню ВС Home I/O розроблена система дозволяє вивчати такі теми автоматизації, як теплова поведінка, керування освітленням, енергоефективність, захист, який забезпечує підсистема сигналізації, – тобто підсистеми, які є частиною повсякденного життя. Основна мета розробленої ВС – ознайомлення з концепціями автоматизації за допомогою інтерактивного «розумного» будинку [46].

Home I/O оснащений найбільш поширеними пристроями автоматизації, проте вимагає розробки сценаріїв керування та розуміння енергетичного

впливу їхнього введення в експлуатацію на системи будинку. LabVIEW можна використовувати для прямого керування Home I/O [46].

В якості альтернативи, Home I/O також може використовуватися як інтерфейс із зовнішніми технологіями, дозволяючи симулятору легко підключатися до різних мікроконтролерних платформ [46].

Подібно до того, як до реальних систем Розумний будинок розробляються мобільні та вебзастосунки, у даній системі представлено вебклієнт для бездротового керування та моніторингу. LabVIEW NXG використовується для бездротового керування та моніторингу. Це можливо завдяки тому, що вся логіка розроблена на панелі діаграм у LabVIEW і вся інформація проходить через один головний ВП, який містить спільні глобальні змінні. У даному ВП додатково було використано бібліотеку System Link для обміну даними з хмарою, до якої звертається програма webVI, розроблена в NXG. Це дає користувачеві можливість отримати доступ до працюючої симуляції «розумного» будинку з будь-якого місця на будь-якому пристрої, який має підключення до Інтернету. Моніторинг і керування симулятором здійснюється через веббраузер в режимі реального часу. Дані з симулятора пересилаються в хмару системних повідомлень через теги читання або запису і навпаки [46].

Отже, ВС Розумний будинок – це гарний приклад того, як ВС у поєднанні з фізичними пристроями можуть бути використані для навчання та проведення досліджень [46].

1.2.9 Порівняння досліджених віртуальних лабораторій та середовищ для навчання в галузі IoT та систем Розумний будинок

Порівняння досліджених VL та ВС для навчання в галузі IoT та систем Розумний будинок представлено у вигляді таблиці 1.1. Їх класифікацію наведено на рисунку 1.15.

Загалом більшість досліджених VL та ВС розроблені для викладачів і студентів, а тому є безкоштовними. Проте мала частина з таких систем надають можливість працювати у групі.

VL пропонують менш комплексні експерименти, зосереджуючись на конкретних завданнях програмування IoT-пристроїв на основі різних мікропроцесорних платформ. Також VL здебільшого представлені як вебклієнти, оскільки такі системи доволі легко реалізувати для використання у веббраузері.

ВС здебільшого представлені як десктопні клієнти, тобто потребують встановлення на ПК користувачів. Проте вони пропонують більші можливості користувачу, здебільшого зосереджуючись на забезпеченні вільного дослідження користувачем віртуального світу та виконанні різних завдань у ньому.

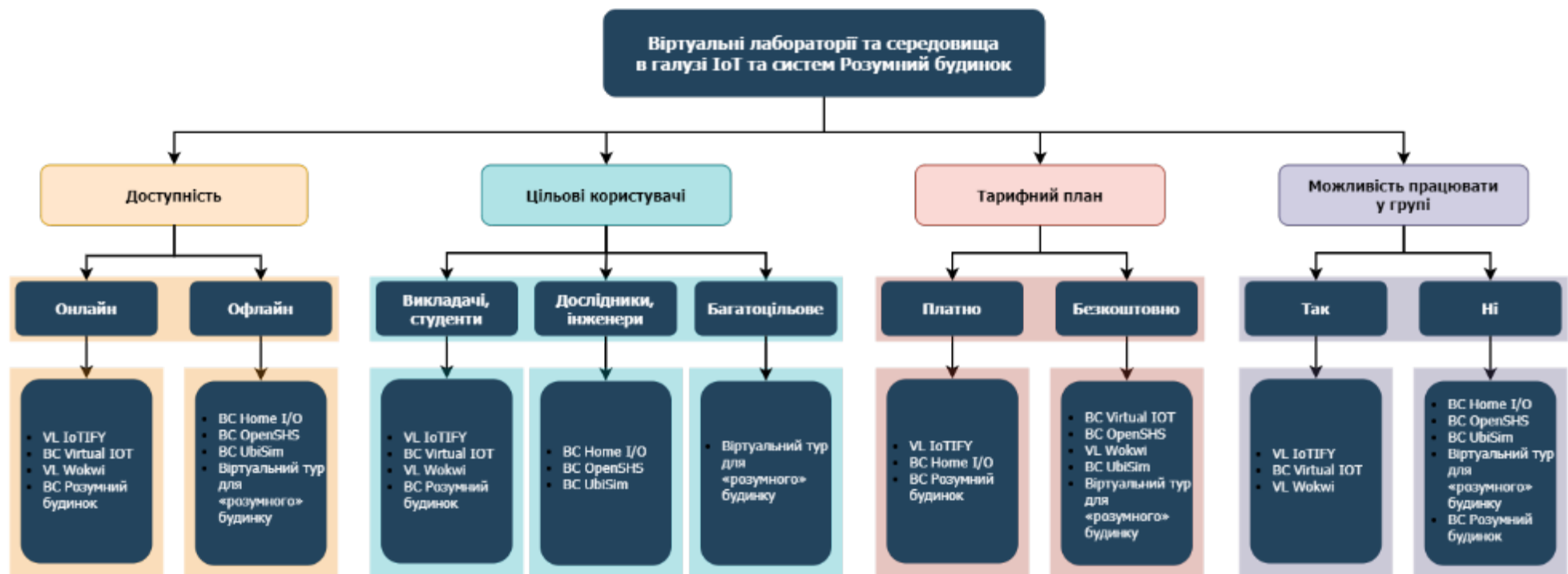


Рисунок 1.15 – Класифікація досліджених віртуальних лабораторій та середовищ для навчання в галузі IoT та систем Розумний будинок

Таблиця 1.1 – Порівняння досліджених віртуальних лабораторій та середовищ для навчання в галузі IoT та систем Розумний будинок

Критерій \ VL, BC	VL IoTIFY	BC Home I/O	BC Virtual IOT	BC OpenSHS	VL Wokwi	BC UbiSim	Віртуальний тур для «розумного» будинку	BC Розумний будинок
1	2	3	4	5	6	7	8	9
Доступність	Онлайн через браузер	Офлайн через клієнт для ПК з можливістю під'єднання пристроїв	Онлайн через клієнт для ПК	Офлайн	Онлайн через браузер	Офлайн	Офлайн	Онлайн
Цільові користувачі	Студенти	Дослідники, викладачі та студенти	Викладачі та студенти	Дослідники та інженери	Студенти, інженери	Дослідники та інженери	Багатоцільовий	Див. BC Home I/O
Експерименти та матеріали	Можна створювати власні проекти з різними вірт. пристроями та сенсорами. Наявні приклади та учбові матеріали	Можна створювати різні сценарії автоматизації. Наявна документація, деякі приклади та підручник для викладачів і студентів	Три рівні з трьома завданнями на кожному. Навчальні відеоматеріали перед кожним завданням	Надається опис системи у [40] та деякі зауваження і відео-інструкції за посиланням [41]	Можна створювати власні проекти. Невелика документація, проте є багато прикладів	Можна створювати власні конфігурації будинків для проведення симуляції	Надає демонстраційні можливості того, як може виглядати взаємодія з «розумним» будинком у майбутньому	Надає можливості для розробки різних сценаріїв керування системою будинку
Тарифний план	Індивідуальна та університетська ліцензія	Індивідуальна, корпоративна ліцензія та безкоштовний пробний термін 30 днів	Безкоштовно	Безкоштовно	Безкоштовно	Безкоштовно	Безкоштовно	Див. BC Home I/O

Продовження таблиці 1.1

1	2	3	4	5	6	7	8	9
Доступні прилади	Мікроконтролери Arduino Uno, STM32 Nucleo F411RE і Raspberry Pi, а також ряд периферійних приладів	174 різних пристроїв, серед яких пристрої освітлення, опалення, контролю доступу тощо	Декілька 3D-моделей мікропроцесорів та приладів	Різні IoT-пристрої (пристрої освітлення, опалення тощо) та меблі	Платформи Arduino UNO, Mega, Nano та деякі інші, багато різних вимірювальних приладів	Датчики руху, тиску, вібрації, температури і контакту, а також RFID-мітки та приймачі	Arduino: різні датчики, п'єзодинамік тощо. BC: вентилятори, жалюзі тощо	Див. BC Home I/O. Пристрій MyRIO з датчиками, ВП
Можливість працювати в групі	+	—	+	—	+	—	—	—
Має відкритий початковий код	—	—	—	+	—	—	—	—

1.3 Аналіз структурних та функціональних особливостей гібридної лабораторії GOLDi

GOLDi (англ. – Grid of Online Lab Devices Ilmenau, укр. – мережа лабораторних онлайн-пристроїв Ільменау) – це гібридна лабораторія, яка пропонує мережу лабораторій по всьому світу і, як наслідок, більше можливостей для ефективного навчання студентів. Апаратне та програмне забезпечення GOLDi було розроблено Групою інтегрованих комунікаційних систем Технічного університету Ільменау, Німеччина. Таке забезпечення також було встановлено в Національному університеті «Запорізька політехніка» та інших вузах у результаті міжнародних освітніх проєктів TEMPUS ICo-op і DESIRE. Сьогодні мережа лабораторій GOLDi об'єднує університети-партнери Німеччини, України, Грузії, Вірменії та Австралії [47]-[48].

Використовуючи GOLDi, студенти можуть розробляти алгоритми керування з різними методами специфікації (опису експерименту) для керування електромеханічними моделями обладнання в ОЛ. Платформу швидкого прототипування системи GOLDi можна використовувати для перевірки та більш ґрунтовного засвоєння навчального матеріалу, отриманого під час прослуховування лекцій з проєктування цифрових систем. Крім цього, можна використовувати спеціальну демонстраційну платформу (м'яч в лабіринті на пластині, яка балансує), щоб дати студентам краще уявлення про можливості та обмеження дистанційного керування і спостереження через Інтернет і критично оцінити ці технології [47].

ГЛ GOLDi пропонує студентам робоче середовище, максимально наближене до умов реальної лабораторії. Крім того, GOLDi мотивує студентів на проєктування систем керування, критично важливих для безпеки [48].

За задумом розробників даної ГЛ технології VL (віртуальні моделі) були вбудовані в інтерфейс RL. Технології VL надають різні можливості для GOLDi, такі, як візуалізація та анімація, що дозволяє спостерігати і тестувати

всі параметри прототипу. У поєднанні з формальними методами проєктування, моделювання та прототипування використовуються для створення основи для розробки надійних архітектур систем. Щоб перевірити функціональність всього проєкту, деякі спеціальні функції моделювання та перевірки було вбудовано як невід’ємну частину системи ГЛ GOLDi. Це надає різні можливості для виконання моделювання, такі, як:

- використання імітаційних моделей фізичної системи для візуального прототипування;
- покрокове і паралельне виконання цих прототипів;
- візуалізація процесу моделювання за допомогою інструментів, які також використовуються для специфікації;
- функції для генерації тестових шаблонів;
- генерація коду для апаратного і програмного забезпечення [48].

GOLDi має розподілену хмарну архітектуру для полегшення процесу підтримки системи кожним університетом-учасником (рис. 1.16). Це дає наступні переваги:

- обслуговування всієї системи в одному центральному місці: www.goldi-labs.net;
- усі університети-партнери мають однакову версію GOLDi;
- нові функції відразу доступні для всіх партнерів;
- використання всіх партнерських лабораторій за допомогою одного центрального облікового запису користувача goldi-labs.net;
- користувач може визначити, яка лабораторія має найменшу затримку (найкращий час реакції) щодо його поточного географічного положення та підключення до Інтернету;
- огляд всіх проведених експериментів у всіх партнерських лабораторіях по всьому світу з можливістю спостерігати за цими експериментами [47].

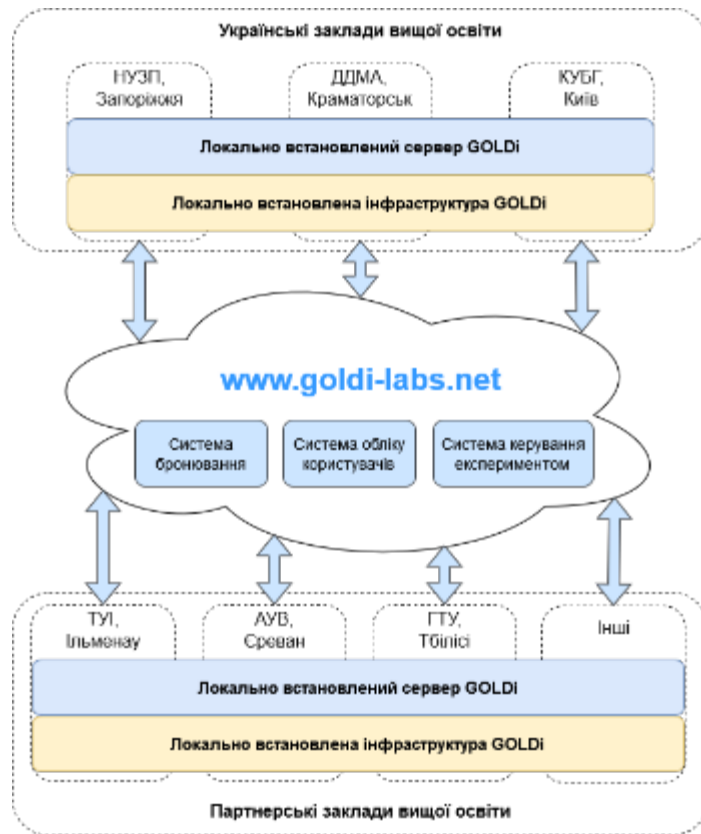


Рисунок 1.16 – Хмарна архітектура ГЛ GOLDi [47]

На рисунку 1.17 показана мережева архітектура системи GOLDi. Сервер (сервер RL, яка налаштовується окремо кожним університетом-учасником) складається з трьох частин:

- внутрішня послідовна шина RL для з'єднання всіх частин RL;
- блок захисту шини (Bus Protection Unit, BPU) для з'єднання блоків керування (БК) з шиною RL і для захисту шини від блокування, неправильного використання та пошкодження;
- блок захисту фізичної системи (Physical System Protection Unit, PSPU), який захищає фізичні системи (електромеханічні пристрої в RL) від навмисного пошкодження або випадкових неправильних команд керування і який пропонує різні механізми доступу і керування [47]-[48].

Взаємозв'язок між блоками вебінтерфейсу (блоками керування) користувача та обраним фізичним пристроєм під час сеансу віддаленої лабораторної роботи (експерименту), а також керування користувачами здійснюється сервером RL, який також керує вебкамерами [47].

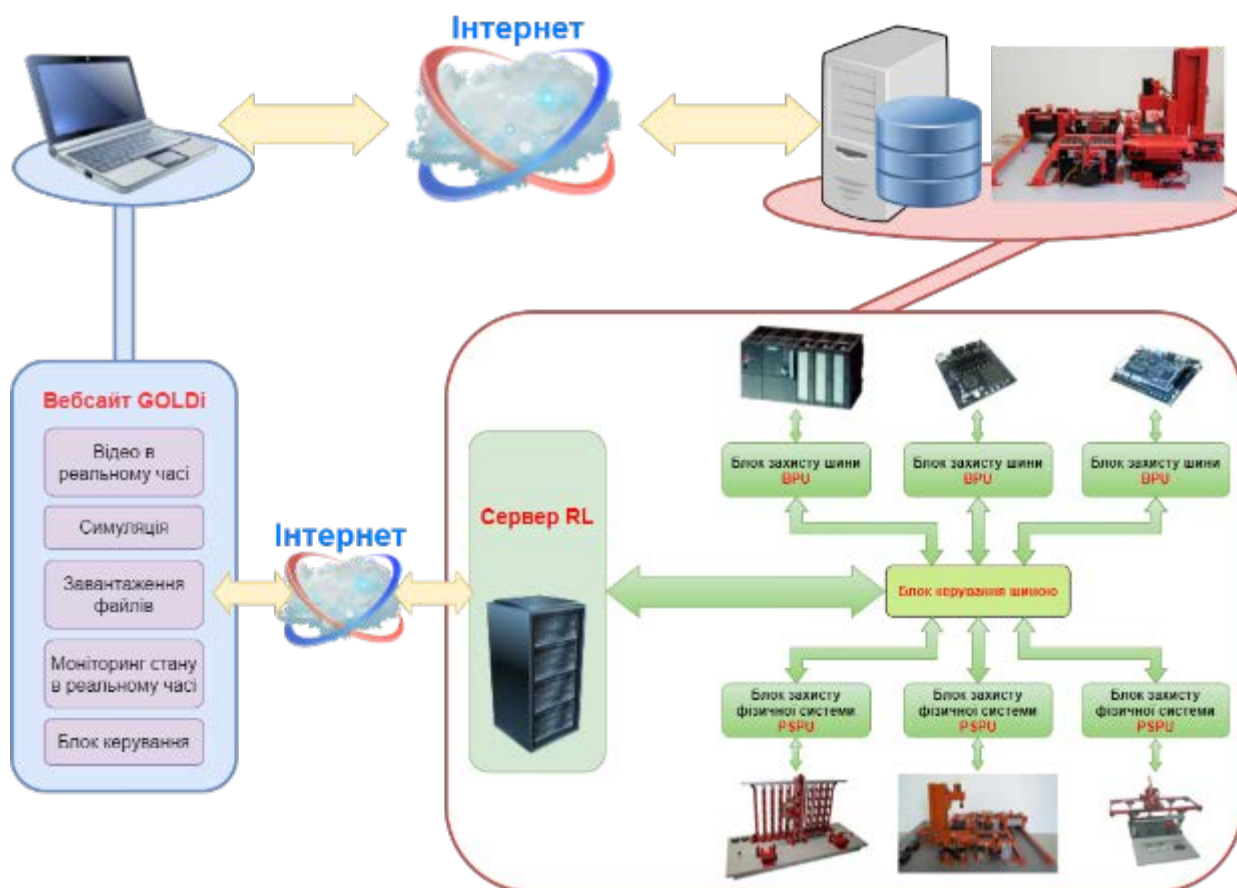


Рисунок 1.17 – Архітектура локального серверу ГЛ GOLDi [48]

Для надійного, універсального і більш гнучкого використання серверу RL було розроблено внутрішню послідовну шину RL. Усі блоки вебінтерфейсу користувача і всі фізичні пристрої з'єднані через цю шину. Завдяки впровадженню даної шини було покращено масштабованість системи, оскільки стало можливим додавання нових вебінтерфейсів керування і нових фізичних пристроїв без значних змін у конфігурації серверу. Це дозволяє створювати внутрішню мережу RL, в якій можна використовувати будь-яку комбінацію фізичних пристроїв та БК для експериментів [47].

BPU отримує команди від БК і перевіряє їх на придатність для обробки шиною. Це робиться шляхом перевірки транспортного протоколу. Зміст переданих даних і адреси не перевіряються, тому що це залежить від використовуваного фізичного пристрою, тому їх перевірка здійснюється відповідними PSPU. Функція BPU полягає в тому, щоб не допустити, щоб БК блокував шину і завдавав шкоду іншим. BPU заснований на тому ж

обладнанні, що і PSPU, але використовує інший набір компонентів для спрощення виробництва і обслуговування. Вторинним завданням BPU є приєднання до шини різних БК, таких, як програмовані логічні контролери. Такий контролер, наприклад, може забезпечувати лише просте керування введенням / виведенням, яке призначене для безпосередньої взаємодії з фізичними системами, тому протокол не реалізований. Серіалізація сигналів введення / виведення і реалізація протоколу виконується завдяки BPU. Перетворення напруги, якщо БК не відповідає напрузі на шині, також виконується завдяки BPU [47].

PSPU дозволяє студентам розробляти еталонний дизайн проєкту, тобто дизайн, який не залежить від використовуваного БК і засобів розробки. PSPU необхідний, коли студенти виконують свої алгоритми безпосередньо на БК і хочуть бути вільними у виборі інструментів проєктування. Завдання PSPU – перевірити безпеку команд шляхом фільтрації всіх команд. Виконуються тільки ті команди, які не можуть призвести до несправності. Всі інші відхиляються і опціонально повідомляються як стан помилки в систему керування навчанням. PSPU може використовуватися з усіма БК, такими, як мікроконтролери, програмовані логічні контролери тощо. Використання такого універсального блоку захисту дає студентам свободу при проєктуванні, оскільки не потрібно брати до уваги запобіжні заходи. Отже, для запобігання збоїв в роботі фізичної системи не потрібно розробляти додаткові механізми безпеки (інструментальні засоби) в межах проєкту для керування програмним і апаратним забезпеченням [47]-[48].

Повний процес проєктування виконується студентами, що дає їм більш реалістичний погляд на реальний процес розробки проєкту. Допустима поведінка PSPU порівнюється з командами, отриманими від БК, щоб визначити будь-які небезпечні команди в розробленому студентом алгоритмі керування. Команди надсилаються на обробку далі тільки тоді, коли вони можливі для виконання на поточному стані фізичної системи. Команди, які є недійсними і можуть викликати збій або будь-яку іншу небажану поведінку,

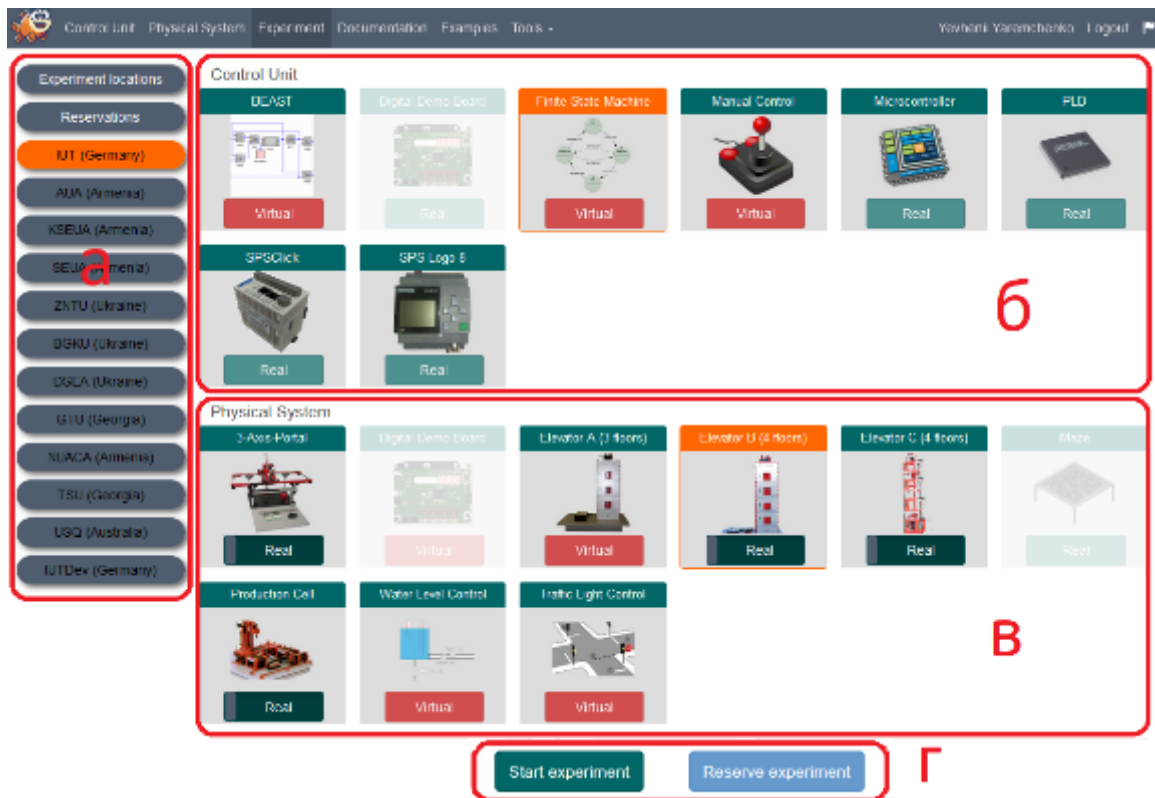
відхиляються і опціонально повідомляються. Крім звичайного вебнавчання для студентів, яке контролюється БК, PSPU також надає можливість доступу вчителів через Інтернет, наприклад, для ініціалізації експериментів або доступу через локальне керування. Також можливо змішувати джерела сигналу, наприклад, для отримання певних сигналів від датчиків та передачі їх через вебінтерфейс. Це можна використовувати у випадках, коли для перевірки поведінки системи потрібне введення даних користувачем на пристрої (наприклад, натискання клавіші ліфту) [47].

На стороні клієнта (ПК студента) GOLDi пропонує вебсередовище – панель керування експериментом (Experiment Control Panel, ECP). ECP підтримує вищезгадані функції для створення і виконання проєкту з використанням імітаційних моделей. Збільшення трафіку бездротового зв'язку і зростаюча кількість мобільних пристроїв, а також сучасні ОТ, такі, як JavaScript, HTML5 і вебсокет створюють нові можливості в галузі дистанційного навчання. Тому для реалізації вебінтерфейсу було обрано HTML5 [47].

На рисунках 1.18 та 1.19 представлено інтерфейс ECP.

Використовуючи ECP, студент може:

- завантажити скомпільовані програми за допомогою відповідних БК;
- обробити експеримент (наприклад, запустити, зупинити, перезавантажити);
- використовувати інтерактивні функції відлагодження (переривання при зміні датчика / виконавчого механізму або за особливих умов);
- за необхідності змінити змінні середовища;
- спостерігати за експериментом, маніпулюючи змінними середовища всередині терміналу введення / виведення або спостерігаючи за маніпуляціями над фізичною системою безпосередньо через вебкамеру [47].



- а) вибір локації (університету) проведення експерименту; б) вибір блоку керування;
в) вибір фізичного пристрою; г) кнопки початку та бронювання експерименту

Рисунок 1.18 – Сторінка вибору експерименту

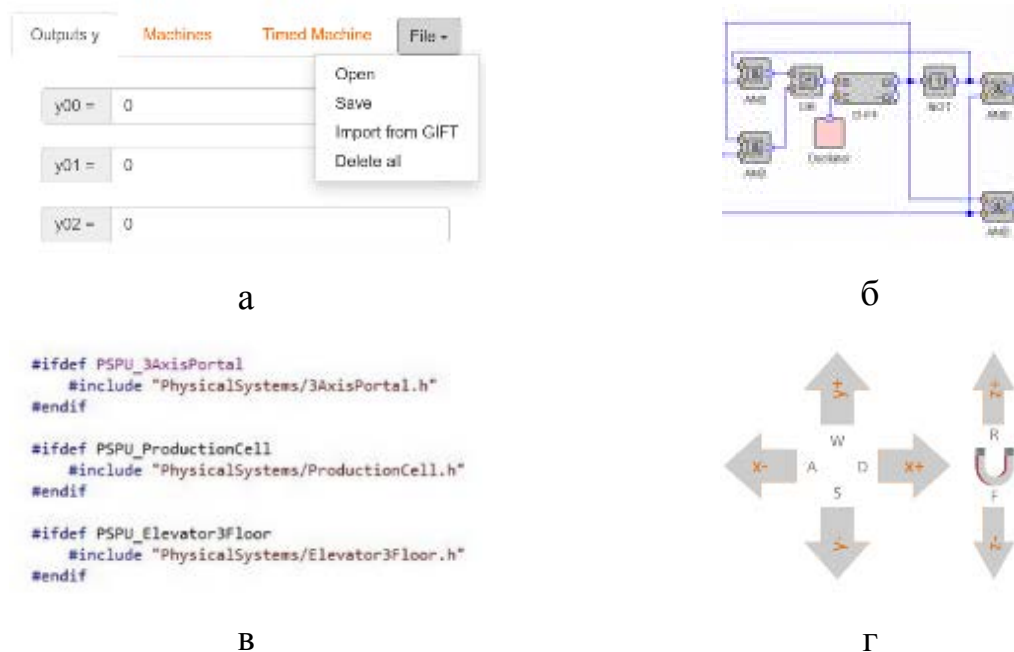


- а) кнопки для завантаження файлів, відображення підказок, відображення актуаторів / сенсорів та зміни налаштувань; б) віртуальна симуляція пристрою; в) зображення віддаленого пристрою в реальному часі; г) блок керування експериментом; г) кнопки для відлагодження, паузи та перезавантаження експерименту

Рисунок 1.19 – Сторінка ECP

Алгоритми керування можуть бути реалізовані у вигляді логічних рівнянь на основі скінченних автоматів, блок-діаграм, ПЗ (наприклад, мовою програмування (МП) Assembler, C) для пристроїв на базі мікроконтролера (FPGA, PLD, SPSClick, SPS Logo 8) або у ручному режимі за допомогою стрілок та кнопок в інтерфейсі ЕСР [49].

На рисунку 1.20 представлено вигляд різних БК ЕСР для реалізації алгоритму проведення експерименту.



а) скінченні автомати; б) блок-діаграми BEAST; в) ПЗ для мікроконтролера; г) ручний режим

Рисунок 1.20 – Вигляд блоків керування ЕСР

Класифікацію експериментів, які надає ГЛ GOLDi можна представити у вигляді схеми, зображеної на рисунку 1.21.

За реалізацією досліджуваного об'єкту експерименти поділяються на віддалені та віртуальні. Слід зауважити, що у разі проведення експерименту з віддаленим пристроєм в ЕСР будуть відображатися і віртуальна симуляція пристрою, і зображення з його камери в реальному часі, а з віртуальним пристроєм – лише його віртуальна симуляція. Експерименти з використанням віртуального обладнання можуть розроблятися на основі тих самих БК, що

використовуються і для роботи з відповідним реальним обладнанням. Віртуальні симуляції можуть мати свої елементи керування, наприклад, кнопки керування ліфтом, зображені на рисунку 1.196 [50].

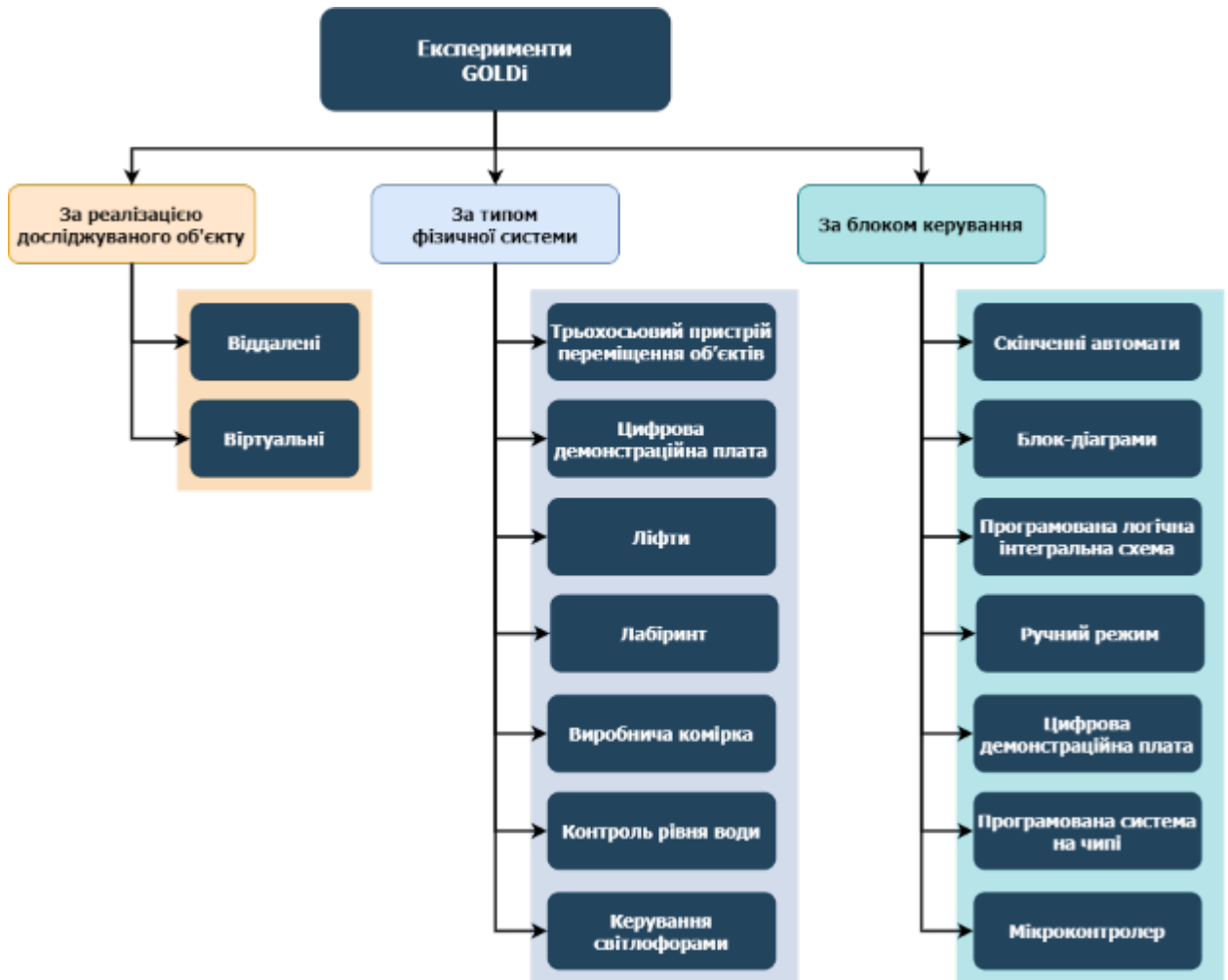


Рисунок 1.21 – Класифікація експериментів ГЛ GOLDi

За БК експерименти поділяються на: скінченні автомати, блок-діаграми, програмовану логічну інтегральну схему, ручний режим, цифрову демонстраційну плату, програмовану систему на чипі та мікроконтролер [50].

Скінченні автомати – це модель поведінки, що складається з скінченного набору станів і переходів між ними. У цьому контексті скінченні автомати описують послідовну поведінку цифрової системи. Залежно від внутрішнього стану система може реагувати на один і той же вхід різними виходами. Для проєктування скінченного автомату використовується

графічний інтерактивний автомат GIFT (Graphical Interactive FSM Tool). Цей інструмент може створювати скінченний автомат у формі скінченного автомата, таблиці переходів, таблиці автомату або z-рівнянь [50].

Блок-діаграми. Експеримент описується за допомогою блок-схем, створених за допомогою редактору BEAST (Block Diagram Editing and Simulating Tool) [50].

Програмована логічна інтегральна схема – це електронний компонент для інтегральних схем. У GOLDi використовується схема MAX V 5M570ZT100C5 від ALTERA. Архітектура цієї схеми об'єднує такі елементи, як флеш-пам'ять, оперативна пам'ять, генератори та контури фазової автосинхронізації тощо. Користувачу потрібно розробити програму для даної схеми, використовуючи ПЗ Quartus II Web-Edition від ALTERA [50].

Ручний режим. За допомогою ручного керування можна керувати обраними для проведення експерименту пристроями за допомогою кнопок, представлених в ЕСР. Дані кнопки можна використовувати для керування переміщенням елементів системи в різних напрямках і для керування деякими конкретними функціями системи в залежності від обраного пристрою [50].

Цифрова демонстраційна плата – це спеціальна плата швидкого прототипування, яка була розроблена для навчальних цілей. Дана плата швидкого прототипування створена на основі схеми MAX V 5M1270Z від ALTERA. Плата має кнопки введення, світлодіодні виходи та інші компоненти. Користувачу потрібно розробити програму для даної схеми, використовуючи ПЗ Quartus II Web-Edition від ALTERA [50].

Програмована система на чипі – це програмована система, яка складається з процесору, цифрових та аналогових блоків та інших функціональних елементів. Даний БК зараз перебуває в процесі розробки, проте взаємодія з ним буде подібною до взаємодії з мікроконтролером [50].

Мікроконтролер – це невеликий комп'ютер на одній інтегральній схемі, що має процесорне ядро, пам'ять і програмовані периферійні пристрої введення / виведення. У GOLDi використовується мікроконтролер Atmega644a

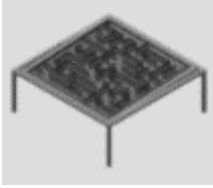
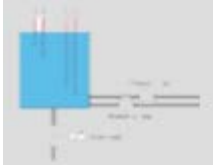
від ATMEL. Цей мікроконтролер має 64 Кб флеш-пам'яті ISP з можливістю читання під час запису, 32 лінії вводу / виведення загального призначення, 32 робочих регістри загального призначення, лічильник реального часу тощо. Користувачу потрібно розробити програму для даної схеми, використовуючи ПЗ AtmelStudio (версія 6.2.x) від ATMEL [50].

Для експериментів доступні декілька моделей фізичних систем, з якими можна проводити експерименти у віддаленому або віртуальному режимі. Опис представлених фізичних систем наведено в таблиці 1.2 [50]-[51].

Таблиця 1.2 – Фізичні системи, доступні для експериментів [51]

Назва	Тип доступу	Зображення	Опис
1	2	3	4
3-Axis-Portal (трьохосовий пристрій для переміщення об'єктів)	Віддалений / віртуальний		Імітує стаціонарний робот-маніпулятор з ортогональним робочим простором, який використовується для передачі заготовок до блоку обробки або сортування на великих виробництвах
Digital Demo Board (цифрова демонстраційна плата)	Віртуальний		Плата швидкого прототипування на базі MAX® V - 5M1270Z CPLD від Altera для навчання основам булевої алгебри, комбінаційної логіки тощо
Elevators (ліфти)	Віддалений / віртуальний		Ліфт складається з клітки з противагами, приямку і декількох поверхів, кожен з яких має розсувні двері, кнопки виклику і кольорові контрольні лампи, які вказують напрямок руху ліфту. Система представлена у декількох конфігураціях – з 3-ома та 4-ма поверхами

Продовження таблиці 1.2

1	2	3	4
Maze (лабіринт)	Віддалений		Демонстраційна платформа, у якій можна переміщувати м'яч в лабіринті на пластині, змінюючи її нахил. Вона дає студентам краще уявлення про можливості та обмеження дистанційного керування і спостереження через Інтернет
Production Cell (виробнича комірка)	Віддалений / віртуальний		Імітує виробничу комірку з вбудованим циркулярним накопичувачем, яка використовується, наприклад, в процесах промислового різання металу
Water Level Control (контроль рівня води)	Віртуальний		Імітаційна модель системи контролю рівня води
Traffic Light Control (керування світлофорами)	Віртуальний		Імітаційна модель системи керування світлофорами

Отже, аналіз функціональних особливостей GOLDi показав, що дана ГЛ пропонує значну кількість віддалених та віртуальних експериментів з різним обладнанням та різними підходами до специфікації експерименту. Проте відсутні засоби для проведення повноцінних навчальних експериментів з розробки систем Розумний будинок та IoT, тому важливою є задача реалізація ВС для проведення таких експериментів. Аналіз структурних особливостей показав, що розподілена хмарна архітектура даної ГЛ полегшує процес підтримки локальних серверів університетами-партнерами та дозволяє додавати як нові пристрої, так і нові блоки керування експериментом без зміни вже існуючих експериментів.

1.4 Постановка задач магістерської роботи

Проведені дослідження показали, що сьогодні існують різні технології онлайн-інженерії, які особливо успішно можуть використовуватися в умовах дистанційного навчання. Такі технології, як VL та BC, доцільно використовувати для створення навчальних систем Розумний будинок та IoT.

Серед досліджених VL та BC може бути вибрано BC UbiSim, віртуальний тур для «розумного» будинку та BC Розумний будинок як аналоги, оскільки у даних системах представлено комплексні рішення, зосереджені на одному користувачеві, процесі його взаємодії з системою та забезпеченні можливості проведення повноцінних експериментів з IoT-систем та систем Розумний будинок [44]-[46].

Проведені дослідження структурних та функціональних особливостей ГЛ GOLDi показали, що це ОЛ зі складною архітектурою, яка є надійним та гнучким інструментом для використання в навчальному процесі в галузі систем автоматизації, в тому числі й систем домашньої автоматизації. Проте у GOLDi відсутня реалізація комплексного експерименту для навчання принципам роботи з системами Розумний будинок, їх програмування та протоколів передачі даних, хоча дослідження показали, що такі VL та BC представлені в Інтернеті. Реалізація подібного BC Розумний будинок може запропонувати студентам нове інтерактивне середовище, яке зможе їх заохотити до активного навчального процесу та вмотивувати на вивчення новітніх технологій та процесів IoT та систем Розумний будинок.

Тому розробка віртуального онлайн-середовища Розумний будинок для гібридної лабораторії GOLDi є актуальною задачею.

Мета роботи – дослідження та програмна реалізація віртуального онлайн-середовища Розумний будинок для розширення функціональних можливостей гібридної лабораторії GOLDi.

Для досягнення мети потрібно вирішити наступні завдання:

- дослідження особливостей існуючих ВС;
- дослідження та розробка методу організації взаємодії користувачів з ВС;
- аналіз технологій реалізації ВС;
- аналіз протоколів передачі даних, що використовуються в системах Розумний будинок та IoT;
- програмна реалізація онлайн ВС Розумний будинок та його тестування;
- розробка методики використання створеного онлайн ВС Розумний будинок.

2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ РЕАЛІЗАЦІЇ ВІРТУАЛЬНИХ СЕРЕДОВИЩ

У цьому розділі для досягнення поставленої мети було проведено дослідження особливостей існуючих ВС, методів організації взаємодії користувачів з ВС, технологій реалізації ВС та протоколів передачі даних для систем Розумний будинок та IoT.

Для проведення цих досліджень було розроблено методологію, яка поєднує наступні етапи:

- пошук наукових робіт за ключовими словами з тем, що були зазначені у завданнях у підрозділі 1.4, за допомогою пошукової системи Google Scholar;
- аналіз технологій, інструментарію та наведених рішень у знайдених наукових працях;
- синтез отриманих результатів аналізу для розробки функціональних та нефункціональних вимог до розроблюваної ВС, методу взаємодії користувачів з ВС, архітектури ВС, а також вибору інструментарію розробки онлайн ВС Розумний будинок з урахуванням необхідної сумісності з HTML5 та JavaScript, які використовуються в ГЛ GOLDi.

2.1 Дослідження особливостей існуючих віртуальних середовищ

Сьогодні ВС застосовуються в різних сферах, наприклад: навчальні симулятори, індустрія комп'ютерних ігор, спілкування, системи моніторингу та керування, виробничі та дослідницькі симулятори, у медицині для реабілітації пацієнтів тощо [52].

Важливим терміном, який часто використовується в контексті розгляду ВС, є «імерсивність». Імерсивність (або занурення) – це термін, який використовується для позначення психологічних станів, які пов'язані з відсутністю усвідомлення часу та реального світу, а також з відчуттям

фізичної присутності у нефізичному світі. Імерсивні технології – це технології, які дозволяють людині відчувати часткове або повне занурення у віртуальний світ [53].

До імерсивних технологій відносять наступні.

VR-шоломи – це пристрої, які імітують природній спосіб отримання людиною зорової інформації, тобто в будь-якому конкретному місці середовище є постійним і користувачі можуть дивитися навколо себе шляхом повороту голови у потрібному напрямку і бачити інформацію, яку вони очікували б побачити у реальному фізичному середовищі. Шоломи такого типу використовують ПК користувачів для рендерингу 3D-графіки та інших розрахунків, оскільки рендеринг та розрахунки потребують великих обчислювальних потужностей. Шолом лише відображає отримані графічні дані. Проте такі шоломи доволі дорогі. Натомість на ринку представлені VR-гарнітури для смартфонів, які виглядають як шоломи-тримачі для смартфонів. У такому випадку на смартфон встановлюється спеціальний застосунок, який розраховує дані та відображає зображення на екрані у вигляді двох кіл, на які користувач дивиться через лінзи шолому і бачить повноцінне зображення. Такі застосунки використовуються для малопотужних 3D-турів [53].

VR-системи – це системи, які пропонують як VR-шолом, так і систему відстеження рухів та виконання певних дій у ВС. Такі системи використовують камери для відстеження рухів та додаткові ручні пристрої, за допомогою яких можна виконувати дії у ВС. Крім цього, VR-системи можуть використовувати різні датчики, які можуть знаходитися на стінах, на підлозі та на користувачах. VR-системи також потребують використання потужного ПК для виконання розрахунків [53].

До неімерсивних технологій належить різне ПЗ для ПК, смартфонів, ігрових консолей тощо, яке відображає створений віртуальний світ на звичайному дисплеї та надає можливості для взаємодії з цим світом за допомогою звичайних пристроїв, таких, як комп'ютерні миші та клавіатури.

Програми-клієнти для ПК можна розділити на дві великі групи: десктопні клієнти (ПК-клієнти) та вебклієнти [27], [53].

ПК-клієнти запускаються в ОС користувача, а тому дуже потужні, оскільки швидкість їх роботи обмежена лише характеристиками ПК користувача. Крім цього, вони мають доступ до нативного функціоналу ОС, а тому можуть надавати функціонал, якого не може дати вебклієнт. Проте вони також менш портатбельні та часто прив'язані до платформи (наприклад, самої ОС у разі ПЗ мовою програмування C / C++ або віртуальної машини Java у разі ПЗ мовою Java). Крім цього, вони менш безпечні для користувача, ніж вебклієнти, оскільки мають доступ до нативного функціоналу ОС. Останнє зауваження особливо важливе для університетських ВС та ОЛ через те, що такі клієнти повинні бути доступні для завантаження лише з сайту університету. У разі завантаження зламаної версії такого ПЗ з іншого сайту ПК студента може бути уражений вірусом, через що університету, можливо, потрібно буде нести певну відповідальність [27].

Вебклієнти працюють у браузері. Вони можуть працювати на будь-якій платформі, що має сумісний браузер, не вимагають встановлення та знаходяться під захистом системи безпеки браузера. Однак певні функції можуть надаватися модулями браузера, що підключаються. Такі модулі можуть бути менш захищеними, ніж сам браузер [27].

Для класифікації ВС можна використати представлені у роботі [54] критерії оцінки RL. Класифікацію ВС можна провести за двома критеріями – технологія реалізації та сфера застосування. На рисунку 2.1 представлено розроблену класифікацію.

Отже, для вирішення поставленого завдання розробки онлайн ВС для ГЛ GOLDi найкраще буде реалізувати його як неімерсійну систему у вигляді вебклієнту, оскільки такий підхід не потребує використання додаткових пристроїв керування, великих обчислювальних потужностей та встановлення ПЗ на ПК користувачів. Тому важливо розглянути методи взаємодії користувачів з ВС та технології, за допомогою яких можливо створити ВС у

вигляді вебклієнту, тобто HTML5-сумісну програму для використання у звичайному браузері.

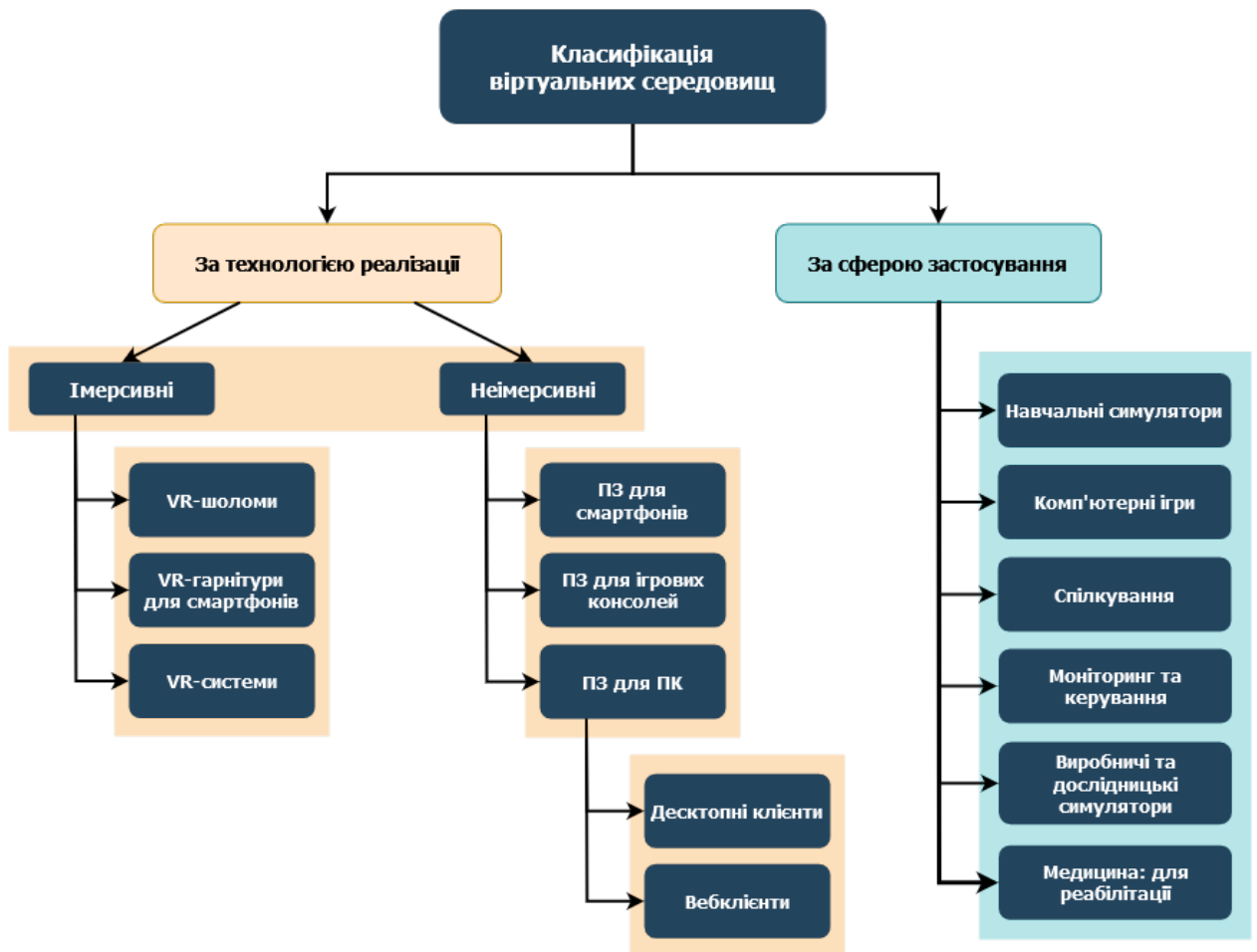


Рисунок 2.1 – Класифікація віртуальних середовищ

2.2 Огляд методів організації взаємодії користувачів з віртуальним середовищем

2.2.1 Поняття «людино-машинна взаємодія»

Людино-машинна взаємодія (ЛМВ) – це сфера досліджень і розробки в галузі проектування та використання комп'ютерних технологій, яка зосереджується на інтерфейсах між людьми (користувачами) і комп'ютерами. Дослідники ЛМВ спостерігають за тим, як люди взаємодіють з комп'ютерами, і розробляють нові технології для покращення їх взаємодії [55].

Важливим поняттям у даній галузі є інтерфейс користувача. Інтерфейс користувача – це подання системи, тобто поєднання всіх її пристроїв введення, концептуальних моделей та методів взаємодії, які використовує користувач. Концептуальна модель – це узгоджена модель, яку користувачі формують щодо функції системи: що це таке, як вона працює та як вона реагуватиме на їхні вхідні дані. Тобто вона дає користувачу розуміння того, які вхідні дані йому потрібно буде надати системі для досягнення бажаного результату. Прикладом концептуальної моделі є метафора папок і файлів на робочому столі. Це концептуальна модель вищого рівня для пояснення того, як інформація зберігається на комп'ютері. UI повинен представляти та посилювати концептуальну модель системи, а також представляти можливості та обмеження, які дають зрозуміти користувачам, як виконати ключові завдання. UI може бути представлений у вигляді GUI, звукових, тактильних та інших інтерфейсів [56].

Метод взаємодії (техніка взаємодії) – це поєднання вхідних і вихідних даних, що надходять з усіх апаратних і програмних елементів, що забезпечує користувачеві спосіб виконання завдання з урахуванням конкретної концептуальної моделі. Наприклад, метод «наведи і клацни» (англ. – Point and Click), який є стандартом для GUI та дозволяє переміщувати та відстежувати покажчик (курсор) поверх об'єктів на екрані, а потім виконувати дії над цими об'єктами в результаті клацання або перетягування миші. Важливо пам'ятати, що сам курсор є одним з можливих елементів методу взаємодії. Методи взаємодії зазвичай відрізняються на різних пристроях введення залежно від їх можливостей сприйняття, здатності легко поєднувати зміни стану, такі, як натискання кнопок, а також фізичних можливостей користувача та зручності під час використання пристрою [56].

Графічний інтерфейс користувача – це один з видів інтерфейсів користувача, який надає їм можливість взаємодіяти з пристроями за допомогою графічних зображень, іконок та інших графічних можливостей. У даному виді інтерфейсу ключовою є парадигма WIMP [55]-[56].

2.2.2 Парадигма WIMP

WIMP («windows, icons, menus, pointer», укр. – «вікна, іконки, меню, покажчик») – це парадигма, яка поєднує багато методів взаємодії користувачів з системами, які мають GUI. Вона була ретельно розроблена більше ніж сорок років тому для мультимодального керування за допомогою клавіатури та миші та здебільшого використовується у ПК. Важливу роль у даній парадигмі відіграє курсор, який є логічною абстракцією фокусу уваги користувача [56].

У даній парадигмі можна виділити декілька головних правил:

- кожне вікно – це окрема програма, тому просте переміщення між вікнами дозволяє легко працювати у декількох програмах одночасно;
- іконки – це графічне подання функцій, які може виконувати пристрій;
- меню поєднують у собі текст та іконки, які разом дозволяють легко обирати потрібну дію з переліку дій, який може залежати від контексту виклику меню [56].

Парадигма WIMP була дуже оптимізована для конкретного сполучення пристроїв введення – миші та клавіатури. Хоча можна керувати курсором і вводити текст за допомогою інших пристроїв введення, зробити це складніше та менш ефективно, ніж за допомогою миші та клавіатури. Багато сучасних платформ покладаються на засоби інтерфейсу WIMP (кнопки, повзунки, прапорці, вікна тощо), водночас забуваючи, що вони оптимізовані для пари пристроїв введення, яких насправді може не бути у складі цих платформ [56].

Як вже було зазначено, WIMP поєднує багато методів взаємодії, а саме: «наведи і клацни», перетягування, вибору та виділення, контекстних меню тощо [55]-[56].

Метод «наведи і клацни». Даний метод є базовим методом взаємодії для графічних інтерфейсів. У даному методі використовується комп'ютерна миша або інший пристрій-показчик, за допомогою якого користувач переміщує курсор на екрані на потрібний об'єкт та клацає на нього (у разі

використання миші найчастіше для цього використовується ліва кнопка). Після цього програма може виконати певні дії на основі даних про місце чи об'єкт, на який клацнув користувач [56]-[57].

Даний метод з різною метою використовується у всіх досліджених у підрозділі 1.2 ВС. Наприклад, у ВС UbiSim користувач може створити модель будинку шляхом натискання кнопки за допомогою миші для вибору елементів, наприклад, дверей, вікон та інших компонентів. Крім цього, у процесі симуляції користувач використовує цей метод взаємодії для встановлення напрямку руху агента та взаємодії з об'єктами [44].

Метод перетягування (англ. – Drag and Drop). У даному методі за допомогою миші або іншого пристрою-показчика користувач обирає потрібний об'єкт, захоплює його затисканням кнопки (наприклад, лівої кнопки миші) та перетягує його в потрібне місце, після чого відпускає затиснуту кнопку. Даний метод дозволяє не лише перетягувати об'єкти, а також організовувати взаємодію між декількома програмами, перетягуючи віртуальні об'єкти з однієї програми в іншу для виконання певних операцій над ними [56]-[57].

Даний метод використовується у ВС UbiSim для створення моделі будинку. Після того як користувач обрав елемент для розміщення у будинку, він перетягує його мишею на потрібне місце. Метод перетягування також активно використовується в процесі розробки моделі будинку у ВС OpenSHS [40], [44].

Метод вибору та виділення. Метод вибору стосується того, як користувачі вказують на певні елементи, щодо яких вони мають намір виконати певні дії. Вибір може виконуватися за допомогою пристроїв-показчиків, прапорців та інших елементів графічного інтерфейсу. Після того як користувач вибрав елемент, використовується метод виділення, який повідомляє користувача про те, що з даним об'єктом зараз відбувається взаємодія, тобто він зараз активний [58].

У досліджених ВС для вибору елементів використовується метод «наведи і клацни». Для виділення активних об'єктів у даних ВС використовуються різні способи підсвічування об'єктів. Наприклад, у ВС UbiSim, якщо агент взаємодіє з датчиком, зона виявлення датчика змінює свій колір на червоний, а сам датчик збільшується (рис. 1.12а). У ВС Розумний будинок елементи мають світлову індикацію (синій, зелений чи червоний контур) [44], [46].

Метод контекстних меню. Контекстне меню – це тип меню, що з'являється за запитом і містить невеликий набір відповідних дій, пов'язаних з елементом керування, ділянкою інтерфейсу, фрагментом даних у програмі або поданням програми. Зазвичай цей контекст задається поточним вибором або іншим чином був визначений користувачем перед викликом контекстного меню [59].

Пропонування невеликого набору відповідних дій у контекстних меню допомагає користувачам знайти саме те, що їм потрібно для виконання поставленого завдання. Цей тип меню також зменшує вартість взаємодії та когнітивне навантаження, оскільки користувачам не потрібно аналізувати довгі списки пунктів головного меню, щоб знайти те, що їм потрібно, і їм не потрібно повертатися до панелі інструментів щоразу, коли вони хочуть використати певну команду. Контекстні меню існують у всіх ОС, як у десктопних, так і в мобільних [59].

Контекстні меню зазвичай не запускаються елементом UI, жестом або взаємодією. Вони з'являються поруч із місцем, де користувач клацає, натискає або проводить пальцем, а результати дії залежать від елемента, з яким взаємодіє користувач [59].

До основних у десктопних ОС взаємодій, які відкривають контекстні меню, належать:

- клацання правою кнопкою миші;
- натискання кнопки Control (Ctrl), а потім клацання;
- клацання двома пальцями на трекпаді [59].

Даний метод активно використовується у ВС UbiSim. У даній системі цей метод використовується для розташування датчиків у будинку. Коли користувач клацає на порожньому місці, з'являється контекстне меню «Розташувати новий датчик», у якому відображається перелік доступних датчиків. Користувач вибирає за допомогою миші датчик та налаштовує його характеристики, такі, як ім'я, радіус виявлення тощо. Крім цього, після натискання у процесі симуляції на об'єкт відображається перелік доступних дій для цього елементу [44].

У ВС Розумний будинок контекстні меню також використовуються для взаємодії з системами будинку. Для цього користувач повинен підійти до об'єкту, який має світлову індикацію, навести на нього курсор та вибрати команду з контекстного меню, яке з'явиться над об'єктом [46].

Як уже було зазначено, парадигма WIMP дуже важлива для систем, які мають GUI. Проте існують і специфічні для ВС методи взаємодії.

2.2.3 Методи взаємодії, специфічні для віртуальних середовищ

Візуальні інтерфейси ВС зазвичай складаються з основної сцени дії, що містить об'єкти, з якими гравець може взаємодіяти (наприклад, аватари, вороги або цілі), і складного рухомого фону (наприклад, інтер'єри, пейзажі). Традиційно на головну сцену дії накладається HUD (heads-up display). Цей дисплей надає контекстну інформацію, пов'язану з поточною ситуацією, у вигляді слів, списків слів, цифр або символів. У більшості випадків більш-менш постійна інформація розміщена у кутах екрану (наприклад, рахунок, рівень здоров'я, карта віртуального світу тощо). Непостійна інформація, навпаки, часто відображається в центрі (наприклад, попереджувальні повідомлення або віджети для виконання дій). HUD змінюється залежно від контексту [60]-[61].

HUD використовується в усіх досліджених у підрозділі 1.2 ВС. Наприклад, у ВС Home I/O він показує карту будинку, напрямок руху,

системний час та різні показники модельованої системи Розумний будинок. У ВС Virtual IOT він відображає рахунок користувача та його інвентар, який він зібрав на карті [36], [38].

Метод спостереження (перспектива спостереження, система камер) за ігровим персонажем ВС визначає правила керування камерою або набором камер для відображення виду 3D віртуального світу та агента у ньому [61]-[62].

Основними методами спостереження є:

- від першої особи: камера знаходиться в перспективі гравця. Користувач бачить віртуальний світ очима ігрового персонажа, ніби користувач сам знаходиться в грі;
- від другої особи: камера з перспективи іншого персонажа. Користувач керує персонажем Б, але бачить його очима персонажа А. Користувач може бачити лише те, що бачить персонаж А;
- від третьої особи: камера дозволяє користувачу повністю бачити персонажа, яким він керує. Камера зазвичай розташована позаду гравця, тому користувач споглядає за ним зі спини у процесі взаємодії зі світом;
- зверху вниз: камера розташована над ігровим персонажем під прямим чи гострим кутом [62].

У більшості з досліджених ВС використовується метод спостереження від першої особи, хоча у ВС UbiSim використовується метод спостереження від третьої особи [44].

Метод переміщення визначає пристрої, за допомогою яких користувач може рухати ігрового персонажа, а також набір ігрових механік, які використовуються для переміщення. До основних механік переміщення належать: рух по прямій лінії вперед, назад, ліворуч чи праворуч (наприклад, розкладка клавіатури WASD), обертання та зміна напрямку спостереження (керування мишею) та стрибки. Проте можуть використовуватися і інші складні механіки, наприклад, біг по стінах [61], [63].

У досліджених ВС використовуються різні методи переміщення. У таких ВС, як Home I/O, Virtual IOT, OpenSHS, віртуальний тур для «розумного» будинку та ВС Розумний будинок використовується клавіатура, за допомогою якої користувач визначає пряму лінію, за якою буде рухатися персонаж у тому напрямку спостереження, у якому дивиться ігрова камера, якою користувач керує за допомогою миші. У ВС UbiSim для переміщення використовується лише мишка, якою користувач задає напрямок руху за методом «наведи і клацни» [36], [38], [40], [44], [45], [46].

Отже, ВС мають специфічні методи взаємодії. Проте у деяких з досліджених у підрозділі 1.2 ВС додатково до традиційного GUI використовується дотикальний інтерфейс користувача.

2.2.4 Дотикальний інтерфейс користувача

Дотикальний (матеріальний) інтерфейс користувача (ДІК) (англ. – Tangible User Interface) – це інтерфейс, в якому користувач взаємодіє з цифровою системою за допомогою маніпуляції з фізичними об'єктами, які пов'язані зі зазначеною системою та безпосередньо представляють характеристики зазначеної системи. Тобто користувач взаємодіє з цифровими даними через фізичне середовище або за допомогою матеріальних об'єктів. ДІК пов'язаний з GUI, оскільки надає інструменти для взаємодії користувача з GUI, який представлений у цифровому вимірі. Проте ДІК може надавати як можливості для введення інформації, так і для її виведення [64].

Найпростішим прикладом такого інтерфейсу є комп'ютерна миша. Перетягування миші по поверхні та відповідне переміщення курсора на екрані є дуже простим способом взаємодії з цифровою системою за допомогою маніпуляції з фізичним об'єктом. Рухи, зроблені за допомогою пристрою, мають чіткий зв'язок із поведінкою, яка запускається в системі (наприклад, покажчик рухається вгору, коли користувач рухає мишу вперед), що робить мишу дуже легким для оволодіння пристроєм введення [64].

У ВС Розумний будинок та віртуальний тур для «розумного» будинку, використовуються додаткові ДІК-пристрої. У віртуальному турі для «розумного» будинку використовується під'єднаний через USB-порт мікроконтролер Arduino з декількома встановленими на ньому датчиками та виконавчими пристроями. Ці електронні пристрої дозволяють отримувати дані про характеристики реального навколишнього світу та відображати їх у ВС і, навпаки, відображати реакцію на події, які відбуваються у ВС. Наприклад, завдяки цьому було реалізовано можливість вмикати / вимикати світло в кімнатах ВС плесканням у долоні. Також є можливість реалізації різних сценаріїв поведінки ВС на основі вимірювання показників реального середовища іншими датчиками пристрою [45].

У ВС Розумний будинок керування будинком можливе з інтерфейсу ВП, створеного у LabVIEW, шляхом фізичного впливу на датчики пристрою MyRIO. Наприклад, зміна рівня освітленості у реальній кімнаті, виміряна завдяки датчикам пристрою, може змінити стан підсистеми освітлення у ВС. У даній системі так само є можливість реалізації різних сценаріїв поведінки віртуального будинку на основі вимірювання датчиками пристрою показників реального середовища [46].

Отже, ДІК – це перспективний вид інтерфейсу, використання якого у ВС може збільшити їх імерсивність [64].

2.2.5 Узагальнення методів організації взаємодії користувачів з віртуальним середовищем

Проведені дослідження показали, що найважливішим типом UI для взаємодії з ВС є графічний. Проте може використовуватися і дотикальний інтерфейс [56], [64].

Графічні інтерфейси ВС використовують дві основні групи методів організації взаємодії користувачів з ВС – WIMP та специфічні для ВС методи. Такі WIMP-методи, як «наведи і клацни» та метод вибору та виділення,

використовуються у всіх досліджених у підрозділі 1.2 ВС. Загалом усі WIMP-методи використовують комп'ютерну мишу як пристрій введення [56].

Специфічні для ВС методи використовуються усіма ВС, оскільки вони є складовими елементами ВС. Досліджені ВС найчастіше використовують метод спостереження за ігровим персонажем від першої особи. Для переміщення більшість ВС використовують поєднання клавіатури та миші [61].

Отже, на основі проведених досліджень запропоновано класифікацію методів організації взаємодії користувачів з ВС, яку представлено на рисунку 2.2. Дана класифікація допоможе розробити метод організації взаємодії користувачів з онлайн ВС Розумний будинок для ГЛ GOLDi.

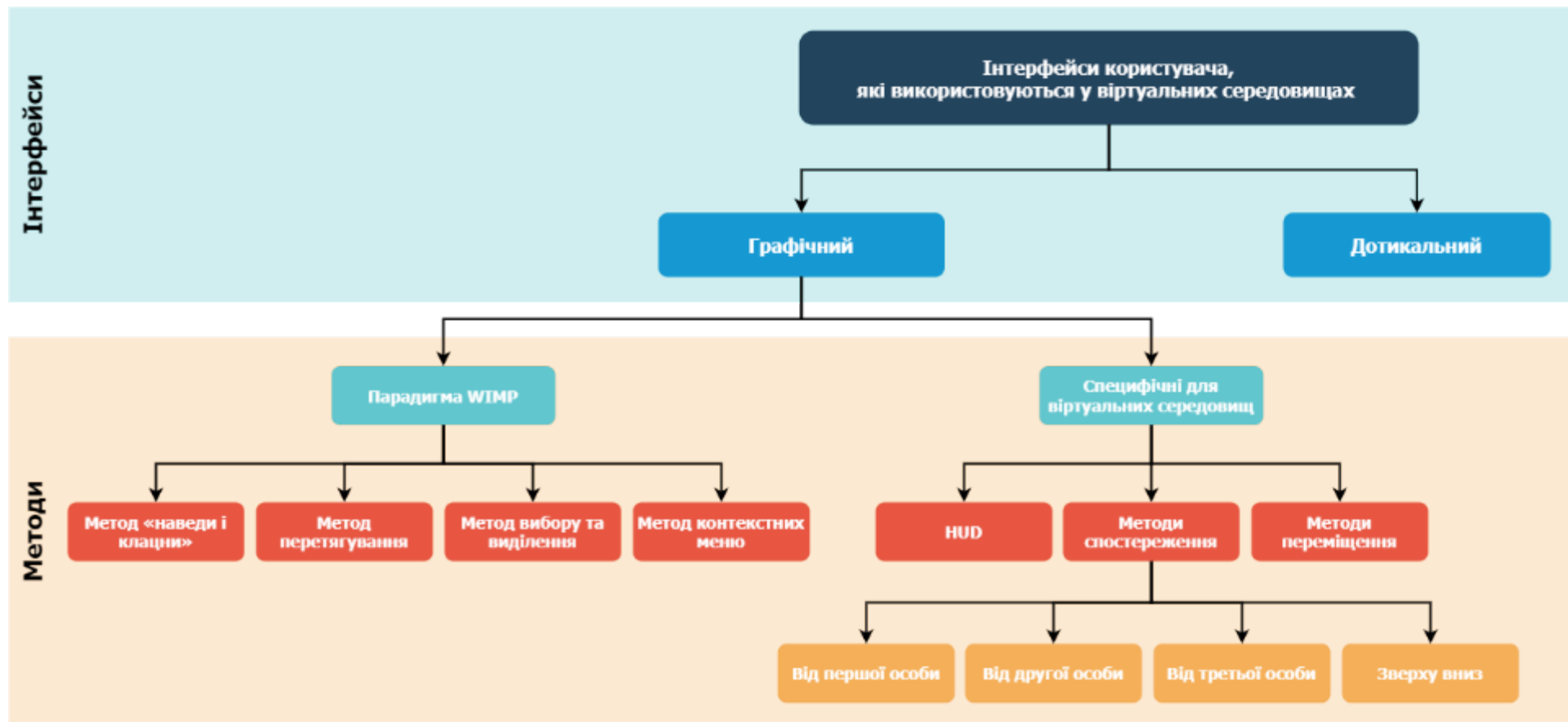


Рисунок 2.2 – Класифікація методів організації взаємодії користувачів з віртуальним середовищем

2.3 Розробка методу організації взаємодії користувачів з віртуальним середовищем

Для розробки методу організації взаємодії користувачів з онлайн ВС Розумний будинок для ГЛ GOLDi доцільно використати GUI та методи взаємодії парадигми WIMP, а саме: «наведи і клацни», вибору та виділення. Це дозволить зменшити кількість потрібних кроків для виконання дій та отримання інформації про ВС Розумний будинок. Вибір елементів буде відбуватися за методом «наведи і клацни», вибрані елементи будуть підсвічуватися та за потреби відображати додаткові інформаційні іконки та віджети [56].

Також потрібно використати специфічні для ВС методи взаємодії:

- HUD, на якому будуть відображатися актуальні дані, повідомлення про помилки, інформаційні віджети та віджети для виконання певних дій;
- перспектива спостереження за ігровим агентом зверху вниз. Хоча використання перспективи від першої особи сприяє більшому зануренню користувача у ВС, проте для проведення експериментів та комплексного дослідження ВС зручнішим є використання перспективи зверху вниз, оскільки користувач може бачити, що відбувається в усіх або майже у всіх приміщеннях ВС. Камера спостереження за ігровим агентом буде знаходитися позаду над ним;
- для переміщення у ВС буде використано лише комп'ютерну мишу, за допомогою якої користувач за методом «наведи і клацни» зможе вказати місце, до якого повинен дістатися ігровий персонаж. Агент зможе виконувати лише прості рухи для того, щоб дістатися цілі, тобто він не матиме можливості пригати [61].

Для взаємодії з об'єктами ВС буде використовуватися комп'ютерна миша та клавіатура. Користувач клацає на об'єкт, який після цього починає підсвічуватися помаранчевим кольором, натискає клавішу F на клавіатурі і,

якщо ігровий агент доволі близько до об'єкту, відбувається певна дія чи відображається віджет, у якому за допомогою графічних кнопок можна виконати певну дію. Віджети матимуть кнопки, текстові поля тощо для виконання дій над об'єктами та будуть використані замість контекстних меню.

Проте метод організації взаємодії користувачів з ВС має бути модифікований шляхом впровадження віддаленого інтерфейсу користувача та методу віддаленого програмування для надання нових засобів отримання даних, підвищення зручності їх обробки та наочності подання.

Віддалений інтерфейс користувача – це інтерфейс, який дозволяє користувачу взаємодіяти з віддаленим середовищем, використовуючи різні методи взаємодії, одним з яких є метод віддаленого програмування. Метод віддаленого програмування – це метод взаємодії користувача з системою, який полягає у використанні мови програмування для надсилання команд системі за допомогою онлайн-технологій. Використання методу віддаленого програмування для взаємодії з системою дозволяє користувачу розробляти зовнішні програми, які можуть контролювати повний комплекс функцій системи. Типовим підходом до реалізації методу є використання TCP/IP-сокетів та будь-якої МП, яка має вбудовані можливості чи реалізовані методи для роботи з ними. У системі, яка реалізує даний метод взаємодії, цільовий пристрій, тобто пристрій, яким можна віддалено керувати, надає API для зовнішніх програм або пристроїв, за допомогою якого вони можуть надсилати команди та отримувати результат. Завдяки цьому методу за допомогою лише одного API можуть підтримуватися різноманітні інтерфейси для розроблюваної онлайн ВС Розумний будинок, наприклад, методи прямих маніпуляцій за допомогою створених студентами власноруч пристроїв з датчиками та актуаторами, ПК, вебклієнтів, смартфонів, ВП тощо [65].

Розроблений метод організації взаємодії користувачів з онлайн ВС Розумний будинок для ГЛ GOLDi можна відобразити у вигляді схеми, представленої на рисунку 2.3.



Рисунок 2.3 – Розроблений метод організації взаємодії користувачів з онлайн ВС Розумний будинок для ГЛІ GOLDi

Практична цінність розробленого методу організації взаємодії користувачів з ВС полягає в тому, що реалізація ВС з використанням запропонованого методу дає інструмент для розвитку практичних навичок програмованої взаємодії з системами Розумний будинок та IoT за допомогою програмно-апаратних клієнтів (ПАК), які користувач розроблює самостійно та може під'єднати їх до розроблюваної ВС для керування ним за методом віддаленого програмування через його API. Така система пропонує високий рівень інтерактивності, оскільки те, що користувач робить у ВС, впливає на стан розроблених ним ПАК і навпаки. Тому важливо розглянути не тільки технології, за допомогою яких можливо створити ВС у вигляді вебклієнту, а й проаналізувати протоколи передачі даних, що використовуються в системах Розумний будинок та IoT, для того, щоб забезпечити можливість віддаленої взаємодії з розроблюваною системою.

2.4 Аналіз технологій реалізації віртуальних середовищ

2.4.1 MATLAB

MATLAB – це комерційна платформа для програмування та числових обчислень, що використовується мільйонами інженерів та вчених для аналізу даних, розробки алгоритмів та створення моделей. MATLAB поєднує в собі інтегроване середовище розробки (Integrated Development Environment, IDE), налаштоване для процесів ітеративного аналізу та проєктування, з високорівневою МП, яка безпосередньо виражає математичні обчислення матриць та масивів. MATLAB має Live Editor для створення скриптів, які об'єднують код, виведення результатів і форматований текст в одну виконувану програму [66].

MATLAB можливо використовувати лише за ліцензією. Щоб використовувати його, навіть за пробною ліцензією, фізичній особі необхідно створити і увійти до свого безкоштовного облікового запису Mathworks. Значні знижки доступні студентам та співробітникам університетів, якщо їхня організація придбала загальносистемну та / або академічну ліцензію. Однак такі корпоративні ліцензії, що дозволяють установку на кількох машинах та ведення журналу з обліковими даними електронної пошти організації, є серйозним вкладенням, і не всі установи можуть собі це дозволити [67].

MATLAB має багато функцій та можливостей:

- аналіз даних: дослідження, моделювання та аналіз даних;
- графіка: різні види графіків для відображення даних;
- розробка алгоритмів для десктопних та вбудованих застосунків;
- розробка як для десктопних платформ, так і для Інтернету;
- використання MATLAB разом з іншими МП, такими, як: Python, C / C++, Fortran, Java тощо;
- підключення до апаратних застосунків;
- розрахунки з залученням потужностей усіх ядер, графічних процесорів та хмарних технологій [66].

Для моделювання в середовищі MATLAB можна використовувати модуль Simulink. Simulink дозволяє проектувати систему з використанням багатодомених моделей, підтримуючи парадигму модельно-орієнтованого програмування. Даний модуль надає можливості для створення 3D-симуляцій (Simulink 3D Animation). Simulink дозволяє створювати великомасштабні симуляції за допомогою компонентів і бібліотек, включно зі спеціалізованими сторонніми інструментами моделювання. Створені симуляції можна запускати на різних платформах. Крім цього, симуляції підтримують багатоядерні процесори та можуть бути запуснені у хмарі [68].

Simulink 3D Animation містить редактори та засоби перегляду для візуалізації та взаємодії з віртуальними сценами. За допомогою 3D World Editor можна імпортувати файли у форматах CAD і URDF, а також створювати деталізовані сцени, які складаються з 3D-об'єктів. Simulink 3D Animation підтримує X3D, стандартний формат файлу ISO та архітектуру часу виконання для представлення та взаємодії з 3D-сценами та об'єктами [69].

Для перегляду створених за допомогою Simulink 3D Animation програм можна використовувати Simulink 3D Animation Web Viewer. Він дозволяє забезпечити доступ до створених віртуальних світів за допомогою будь-якого веббраузера з підтримкою HTML5 та JavaScript. Створений у Simulink 3D Animation віртуальний світ можна запустити на головному комп'ютері, а потім переглянути його у веббраузері на іншому ПК. Для цього не потрібно встановлювати Simulink 3D Animation або інші плагіни на віддаленому ПК [70].

Для розміщення ПЗ, створеного за допомогою MATLAB та Simulink, в Інтернеті у вигляді вебсайту пропонується використовувати компонент платформи MATLAB Web App Server. Для його використання розробник повинен встановити MATLAB та його компоненти – MATLAB Compiler та MATLAB App Designer. Тільки програми, розроблені за допомогою MATLAB App Designer, можуть бути розгорнуті як вебзастосунки на сервері MATLAB Web App Server. За допомогою MATLAB App Designer можливо створювати

ПЗ з GUI, яке можна скомпілювати за допомогою MATLAB Compiler та розмістити його в Інтернеті за допомогою MATLAB Web App Server. Кінцеві користувачі можуть використовувати такі застосунки за допомогою звичайного веббраузера без необхідності встановлення додаткових ПЗ чи браузерних плагінів [71].

MATLAB Web App Server підтримує інтеграцію зі стандартами автентифікації, такими, як OpenID Connect та LDAP, для забезпечення контролю доступу до розроблених вебзастосунків. Крім цього, можливим є розміщення та спільне використання кількох програм, розроблених з використанням різних версій MATLAB та Simulink [71].

MATLAB Web App Server розроблений для роботи лише у надійному середовищі інтрамережі, а не у відкритому Інтернеті. Сервер може працювати на платформах Windows, Linux та MacOS [71].

Також у роботі [72] представлено підхід до реалізації вебклієнту, який не потребує використання MATLAB Web App Server. У розроблений RL, у якій роль серверу покладається на модуль Simulink, який взаємодіє з пристроєм через USB, взаємодія між вебклієнтом та сервером реалізована за допомогою вебсокетів. Вебклієнт створений за допомогою HTML5, CSS та JavaScript. Такий підхід спрощує розробку вебклієнтів ВС та ОЛ за допомогою MATLAB та Simulink.

Приклад реалізації у вигляді ВС простої системи круїз-контролю автомобіля, представлений у роботі [73], показує, що рівень графіки створених за допомогою MATLAB Simulink 3D Animation симуляцій доволі низький, оскільки використовується звичайна векторна графіка без застосування графічних ефектів.

2.4.2 Unity

Unity – це 2D / 3D ігровий рушій (Game Engine, GE) і потужне IDE для розробки ігор. Як GE Unity надає найважливіші функції для розробки ігор,

такі, як: фізика (виявлення зіткнень, визначення сил, що діють на об'єкти під час гри, розрахунок перетинань тощо), 2D- та 3D-рендеринг тощо. Основною МП є C#, а для пришвидшення процесу розробки можна використовувати систему візуального програмування Bolt [74].

Ігровий рушій – це середовище розробки ПЗ, в основному призначене для створення відеоігор. Основні функції GE полягають у забезпеченні засобів рендерингу 2D- та 3D-графіки, фізичного рушія, анімації, штучного інтелекту, звукового рушія, взаємодії з мережею Інтернет тощо. Це платформа, яка використовується для створення різних типів відеоігор за допомогою різних МП. GE подібний до IDE, але має багато специфічних налаштувань, що дозволяють легко розробляти ігри [75].

Протягом багатьох років основна увага розробників Unity була зосереджена на демократизації розробки ігор, щоб кожен бажаючий міг написати гру та зробити її доступною найширшій аудиторії. Однак жоден програмний пакет не може ідеально підходити для всіх ситуацій. Unity добре підходить у таких ситуаціях:

- під час створення гри для декількох цільових платформ. Кросплатформність Unity є однією з найкращих в індустрії, і якщо необхідно створити гру, що буде доступна на кількох платформах (десктопних, мобільних та веб), Unity може виявитися найкращим вибором у такій ситуації;
- коли важлива швидкість розробки. Можна витратити місяці на розробку GE, що має всі необхідні функції, або можна використати сторонній рушій, наприклад, Unity чи Unreal Engine 4;
- коли потрібен повний набір функцій, але немає бажання створювати власний комплект інструментів [76].

Проте в деяких ситуаціях Unity виявляється не таким корисним:

- в іграх, що не потребують частого перемальовування сцени. Unity гірше підходить для реалізації ігор, які не потребують інтенсивних операцій з графікою, оскільки Unity перемальовує кожен кадр. Це необхідно для

підтримки анімації в реальному часі, але потребує більшої кількості розрахунків;

- коли потрібне точне керування процесами рушія. Відмовившись від придбання ліцензії на вихідний код Unity (така можливість є, але вона використовується рідко), стає неможливим контролювати поведінку рушія на низькому рівні [76].

Unity доступний для Windows, MacOS та Linux і поширюється за такими ліцензіями: Personal, Plus, Pro та Enterprise. Вони мають такі властивості:

- версія Personal призначена для індивідуальних розробників, які бажають використовувати Unity для створення власних ігор. Версія Personal надає базовий функціонал та доступна безкоштовно для тих розробників, у яких обсяг залучених інвестицій не перевищує 100 тис. доларів на рік. За використання даної ліцензії до гри додається екран-заставка з логотипом Unity (у всіх інших ліцензіях така умова відсутня);

- версія Plus призначена для індивідуальних розробників та невеликих команд. У цій версії відкривається доступ до засобів автоматичного збору звітів про помилки в роботі гри під час використання на пристроях користувачів. Вона доступна за 399 доларів на рік для тих розробників, у яких обсяг залучених інвестицій не перевищує 200 тис. доларів на рік;

- версія Pro надає доступ до всіх можливостей Unity. Вона доступна за 1800 доларів на рік для тих розробників, у яких обсяг залучених інвестицій перевищує 200 тис. доларів на рік;

- версія Enterprise рушія Unity, призначена для великих команд та надає доступ до всіх можливостей Unity, сервісів підтримки користувачів Unity та онлайн-семінарів. Вона доступна за 4000 доларів для 20 ПК [77].

Unity WebGL дозволяє відображати 2D- та 3D-графіку у веббраузері без необхідності встановлення додаткових плагінів на ПК кінцевих користувачів. Unity WebGL створює HTML5 / JavaScript програму, яку можна розмістити на вебсайті [78].

Отже, GE Unity дозволяє розробляти реалістичні віртуальні світи, що було добре показано на прикладі BC Home I/O [36] та BC, представленої у роботі [79]. Колаборативне BC для навчання співробітників служби безпеки під час великих подій, запропоноване в роботі [79], відтворює реальний сценарій навчальних дій з високим ступенем інтерактивності та занурення. Розроблене за допомогою Unity BC, яке надає інтерактивні ресурси, що були розроблені за допомогою Autodesk 3ds Max, показало свою ефективність для виконання тренувальних завдань, що значно підвищило ефективність підготовки співробітників.

2.4.3 Unreal Engine 4

Unreal Engine 4 (UE4) – популярний потужний ігровий рушій, четверте покоління технології рушія Unreal Engine від компанії Epic Games, розвиток якої почався в середині 1990-х років. Як GE він забезпечує загальні функції, які використовуються в розважальному ПЗ, включно з системою рендерингу тривимірної графіки, системою програвання аудіо, керуванням пристроями введення, мережевою інфраструктурою, підтримкою штучного інтелекту тощо. Він забезпечує можливість розробки для безлічі платформ, серед яких настільні та мобільні ОС, ігрові консолі, VR-пристрої та веб. UE4 також стає все більш популярним для використання у кіно та архітектурі для створення рекламних роликів, різних сцен, моделювання інтер'єру приміщень тощо [80].

UE4 використовується для створення ігрових сцен, розміщення на них об'єктів, написання логічних та математичних операцій, налаштування освітлення та матеріалів (текстур), а також остаточного складання програми. Переваги його використання:

- надання точної моделі освітлення дозволяє досягти реалістичного зображення;
- надання спеціальних редакторів моделей, текстур, матеріалів, поведінки штучного інтелекту, анімації тощо;

- система візуального програмування Blueprint, що прискорює процес розробки та усуває необхідність знання МП;
- великий набір засобів розробки та безкоштовних плагінів;
- надання розробнику готових шаблонів з моделями, текстурами та матеріалами, на основі яких можна створити власну гру;
- відкритий доступ до початкового коду рушія, що дає можливості для його покращення та адаптації під конкретні потреби розробника, що особливо важливо для розробників великих ігор [81].

UE4 зробив революцію в методах оплати розробниками ліцензій на використання GE. Замість того, щоб витратити гроші на придбання ліцензії для використання рушія, розробники можуть використовувати його безкоштовно та без обмежень отримувати доступ до всіх його можливостей, якщо вони не прагнуть його використовувати для розробки проєкту з метою отримання прибутку або якщо їх прибуток за весь термін продажу їх гри становить менше, ніж один мільйон доларів. Після отримання прибутку більше одного мільйону доларів, розробник повинен сплачувати 5% від прибутку щорічно. Крім цього, для великих компаній існує можливість придбання ліцензії на використання рушія на одному ПК за 1500 доларів на рік або підписання договору ліцензування на інших умовах [82]-[83].

UE4 пропонує великий набір інструментів розробки AAA-ігор (умовне позначення ігор, які створюються та розповсюджуються великими видавництвами) з передовими технічними характеристиками для всіх, хто прагне розробляти відмінні ігри. GE був розроблений на C++ і забезпечує підтримку ООП як під час розробки гри за допомогою C++, так і в разі розробки за допомогою Blueprint. Це разом з відкритим початковим кодом рушія забезпечує масштабованість розроблюваної гри. Проте використання МП C++ через її складність знижує темпи розробки гри [82].

Реалізація HTML5 у UE4 використовує набір інструментів Emscripten від Mozilla для кроскомпіляції коду UE4 з C++ у JavaScript. Коли UE4 завантажується з Epic Games Launcher або репозиторію Epic на сайті Github,

всі необхідні файли Emscripten і Python для розробки HTML5 будуть також завантажені та встановлені, а тому можна буде одразу почати створювати проєкти для платформи HTML5. У разі компіляції UE4 самостійно з його початкового коду потрібно впевнитися, що необхідні компоненти також було встановлено. Проте створення проєктів для платформи HTML5 можливе до версії UE4 4.24 включно (актуально версія 4.27) [84].

Emscripten – це повний набір інструментів компілятора для трансляції коду, написаного за допомогою МП С, С++ тощо, у WebAssembly (кросплатформний низькорівневий проміжний код для браузерів), що використовує LLVM (компілятор, який в результаті створює WebAssembly-код). Emscripten створений з особливою увагою до швидкості виконання, розміру і вебплатформи. Emscripten конвертує OpenGL у WebGL та підтримує знайомі API, такі, як SDL, pthreads та POSIX, а також веб-API та JavaScript [85].

У роботі [86] представлено ВС на основі UE4 для навчання технологіям промислового в'язання. Відгуки студентів щодо розробленої за допомогою UE4 системи показують, що тривимірна віртуальна система навчання технологіям промислового в'язання зручніша і більш захоплююча, оскільки завдяки якісній реалістичній 3D-графіці студенти відчувають так, ніби використовують справжню машину на справжній в'язальній фабриці.

Отже, GE UE4 – це потужний рушій, який дозволяє розробляти фотореалістичні віртуальні світи з високим рівнем інтерактивності та імерсивності, що навіть дозволяє використовувати його у різних сферах за межами розробки комп'ютерних ігор [80].

2.4.4 HTML5, WebGL, JavaScript

HTML5 (англ. – HyperText Markup Language, укр – мова розмітки гіпертексту) надає розробникам нові функції, такі, як анімація, можливості кешування для автономного зберігання даних, програвання аудіо та відео, збільшення графічних можливостей та багато іншого, що дає новий клас

вебстандартів та скасовує потребу в пропрієтарних технологіях, наприклад, Adobe Flash [87].

Розвиток HTML5 контролюється організацією W3C (організація з контролю розвитку Інтернету), членами якої є веброзробники, розробники браузерів, представники різних академічних кіл та ін. Це означає, що будь-хто може використовувати дану технологію для розробки різного вебконтенту і, як наслідок, забезпечити еволюцію вебконтенту від статичного текстового контенту до інтерактивного 3D-контенту. HTML5 – це набір великої кількості нових API-інтерфейсів і семантики для веброзробників, які покращують представлення вебконтенту, підвищують гнучкість і портабельність застосунків. Однією з найважливіших функцій, представлених у HTML5, є тег canvas, який дозволяє легко інтегрувати мультимедійний контент без використання будь-якого стороннього ПЗ та плагінів. За допомогою цього тега 2D- та 3D-контент може бути вбудований у вебсторінку [87].

Деякі інші нові функції та API було представлено в HTML5:

- WebWorkers – це набір методів, що використовуються для запуску вебскриптів у фоновому режимі, що забезпечує багатопоточність функцій публікації запитів та відповіді на них;
- WebSockets, що забезпечують двосторонній зв'язок між клієнтом та сервером;
- CSS3 – вдосконалена версія CSS2 з низкою нових підходів до градієнтів та можливостей анімації вмісту;
- швидший рушій JavaScript з розширенням для роботи з рушієм OpenGL [87].

WebGL – одна з найбільш розвинених бібліотек, яка підтримується специфікацією HTML5 та реалізується через тег canvas. Бібліотека містить набір класів та методів, які розширюють відому МП JavaScript для надання можливості створення та рендерингу 3D-контенту в мережі. Бібліотека WebGL використовується як інтерфейс стандарту OpenGL для МП JavaScript. Завдяки цьому JavaScript дозволяє виконувати код OpenGL, використовуючи

можливості відеокарти. Прямий доступ до апаратного забезпечення, тобто відеокарти ПК, забезпечує більш високу продуктивність під час рендерингу 3D-контенту порівняно з рендерингом 3D-графіки за допомогою ПЗ, яке для виконання графічних розрахунків використовує центральний процесор ПК. ОС обробляє запити WebGL за допомогою набору бібліотек OpenGL, встановлених у системі [87].

WebGL не вимагає від користувачів встановлення додаткових плагінів, щоб переглядати та використовувати 3D-контент на сайті. Єдиною умовою є те, що веббраузер користувача повинен підтримувати HTML5 та тег canvas [87].

Для розробки браузерних вебзастосунків з використанням 3D-графіки було розроблено безліч JavaScript-фреймворків та бібліотек, наприклад:

- Three.js, Babylon.js та Zdog – це високорівневі кросбраузерні бібліотеки для рендерингу 3D-графіки за допомогою WebGL;
- Aframe – це JavaScript-фреймворк з відкритим кодом для створення VR в Інтернеті за допомогою WebGL;
- PlayCanvas та WhitestormJS – це ігрові рушії з відкритим початковим кодом. Вони використовують HTML5 та WebGL для запуску ігор та іншого інтерактивного 3D-контенту у всіх сучасних браузерах без використання плагінів;
- Xeogl – це 3D-рушій на основі WebGL для технічної візуалізації;
- Cannon.js – це фізичний рушій, який реалізує прості закони фізики твердого тіла, а саме: просте виявлення зіткнень, різні форми тіла, їх взаємодію, тертя тощо [88].

Нові можливості HTML5 та WebGL дозволяють створювати повноцінні онлайн 3D-світи у веббраузері. Katalabs – це невелика компанія, що вийшла з дослідницької спільноти Стенфорда. Вона створила платформу з відкритим початковим кодом для створення віртуальних світів у браузері. ПЗ KataSpace використовує нові вебстандарти, такі, як WebGL і WebSockets. Воно дозволяє користувачам взаємодіяти один з одним у 3D-середовищі.

Користувачі можуть мати власні аватари, рухатися за допомогою стрілок на клавіатурі та спілкуватися один з одним у чаті [89].

Отже, HTML5, WebGL та JavaScript – це доволі прості інструменти, які дають можливість створювати ВС прямо для браузера без необхідності писати код іншою МП та подальшої його компіляції спеціально для використання у веббраузері [87].

2.4.5 Порівняння технологій реалізації віртуальних середовищ

Отже, як показав проведений аналіз, для розробки ВС може бути використано різні технології. Проаналізований перелік технологій не є вичерпним, оскільки для розробки віртуальних 3D-світів загалом може бути використано різні GE та системи для моделювання інтерактивних 3D-систем. Окрім інструментарію, описаного в пунктах 2.4.1-2.4.4, з цією метою можуть бути використані наступні системи, які були використані для розробки ВС, описаних в наукових працях:

- представлене у роботі [90] ВС для оцінки можливостей виконання процедур безпеки при аматорському підводному плаванні з аквалангом створено на базі мультимедійної платформи Adobe Flash. Дана платформа надає можливості для розробки векторної графіки, анімації, ігор тощо, які можна переглядати за допомогою веббраузера. Дана технологія потребує встановлення браузерного плагіна Adobe Flash Player. Проте платформа Adobe Flash вже застаріла і її підтримку та розробку було повністю припинено 31-ого грудня 2020 року;

- наведений у роботі [91] симулятор польотів використовує розширення для браузерів Microsoft Silverlight. Дана технологія конкурувала з Adobe Flash і вона також є застарілою. Підтримку Microsoft Silverlight було повністю припинено 12-ого жовтня 2021 року.

Порівняння існуючих систем та інструментарію з підтримкою HTML5 та JavaScript для реалізації ВС представлено у вигляді таблиці 2.1.

Таблиця 2.1 – Порівняння технологій реалізації віртуальних середовищ

Критерій \ Інструмент	MATLAB (Simulink)	Unity	Unreal Engine 4	WebGL, JS-фреймворки
1	2	3	4	5
Платформа, на якій працює	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows, Linux, MacOS	Усі веббраузери, які підтримують HTML5
Платформа, для якої може зібрати проєкт	ОС, на яких працює середовище розробки, веб	ОС, на яких працює GE, мобільні ОС (Android, iOS), ігрові консолі, VR-пристрої, веб тощо	ОС, на яких працює GE, мобільні ОС (Android, iOS), ігрові консолі, VR-пристрої, веб тощо	—
Графічні можливості	Здебільшого 2D, але є 3D-можливості, які не забезпечують високого рівня реалізму	Високий рівень рендерингу 3D-графіки, світла та графічних ефектів, що вкупі забезпечують високий рівень реалізму	Відмінна 3D-графіка, що використовується навіть у кіно. Це досягається завдяки багатьом налаштуванням світла, завдяки яким можна швидко налаштувати необхідне освітлення та ефекти	Доволі високі можливості рендерингу 3D-графіки, проте для його програмування може знадобитися багато часу
Мова програмування	MATLAB, Python, C/C++, Fortran, Java	C#, JavaScript, UnityScript, Bolt	C++, Blueprint	JavaScript, TypeScript
Тарифний план	Лише платні ліцензії: різні види комерційної ліцензії, ліцензія для закладів освіти	Безкоштовна ліцензія Personal та платні версії Plus, Pro та Enterprise	Безкоштовне необмежене використання, платити потрібно лише в разі прибутку більше 1 мільйону доларів. Існують інші договірні ліцензії	Безкоштовно, але деякі фреймворки та бібліотеки можуть бути платними
Цільові користувачі	Вчені та інженери	Розробники малих та середніх ігор	Великі компанії-розробники ігор, архітектори, кінокомпанії	Веброзробники
Особливості	Надає потужний інструмент для математичних розрахунків	Добре підходить для розробки малих та середніх ігор	Використовується для розробки великих ігор та у кіноіндустрії завдяки своїм графічним можливостям. Добре підходить для онлайн-ігор, оскільки пропонує великі можливості для роботи з мережею	Використовує стандартні можливості HTML5 і не потребує обов'язкового використання IDE для написання коду

Продовження таблиці 2.1

1	2	3	4	5
Необхідність встановлення додаткового ПЗ на ПК кінцевих користувачів	—	—	—	—
Документація	Велика документація та багато різних прикладів	Велика документація та багато різних прикладів	Багато різних прикладів, безкоштовних навчальних курсів, але документація не є вичерпною	Велика документація, багато різних прикладів та безкоштовних навчальних курсів
Розширюваність плагінами та бібліотеками	+	+	+	+
Надає відкритий доступ до початкового коду	—	Тільки з ліцензією Pro та Enterprise	+	Здебільшого всі бібліотеки є відкритими
Спільнота розробників	Активна, є офіційний форум	Активна, є офіційний форум	Активна, є декілька офіційних форумів	Активна, є багато форумів, розробляється під наглядом W3C

У результаті можна зробити висновок, що найкращим підходом для реалізації ВС є використання GE, оскільки можливості рендерингу 3D-графіки та рівень інтерактивності створених у MATLAB програм не задовольняє вимог до ВС, а використання WebGL та JavaScript-фреймворків з метою створення ВС потребує багато часу на програмування не тільки логіки взаємодії об'єктів 3D-світу, а й програмування їх відображення.

Ігрові рушії Unity та Unreal Engine 4 є потужними системами для створення ігор та подібні за багатьма критеріями. Вони пропонують високий рівень графіки (хоча можливості UE4 у цьому значно більші, проте такі графічні можливості частіше використовуються під час розробки AAA-ігор, моделювання інтер'єрів дизайнерами та архітекторами, а також у кінематографі), обидва пропонують можливості для візуального

програмування та розробки для вебсередовища (можуть скомпілювати проєкт в HTML5 / JavaScript програму). Проте UE4 надає більше можливостей для роботи з мережею, що важливо для розробки онлайн ВС, та пропонує шаблони ігор з моделями, текстурами та матеріалами, які можна використати для розробки власного проєкту. Крім цього, UE4 надає доступ до всіх можливостей рушія безкоштовно (оскільки розроблюване ВС для GOLDi не має комерційної мети) та не додає свій логотип у скомпільовану програму [92].

Отже, для розробки онлайн ВС для ГЛ GOLDi доцільно обрати GE Unreal Engine 4.

2.5 Порівняння протоколів передачі даних, що використовуються в системах Розумний будинок та IoT

2.5.1 Класифікація протоколів передачі даних

Протоколи є невід'ємною частиною стеку технологій, які використовуються в системах Розумний будинок та IoT. Стек протоколів у межах мережевої моделі OSI (англ. – Open Systems Interconnection Basic Reference Model, укр. – базова еталонна модель взаємодії відкритих систем), які найчастіше розглядаються в контексті технологій систем Розумний будинок та IoT, представлені на рисунку 2.4 [93].

Протоколи прикладного рівня (IETF CoAP, IBM MQTT, XMPP та AMQP) використовуються для підключення Розумних речей, а також ПАК кінцевих користувачів до Інтернету. Протоколи транспортного рівня (UDP та TCP) призначені для передачі даних між ПАК. Протоколи мережевого рівня (IPv6, ROLL RPL та 6LoWPAN) забезпечують можливості мережної маршрутизації та інкапсуляції каналів зв'язку. BLE, Z-Wave, ZigBee, HomePlug GP та Dash7 – це п'ять стандартних протоколів канального рівня з низьким енергоспоживанням для бездротового зв'язку на короткі відстані у межах однієї локальної мережі. На фізичному рівні основними протоколами є

IEEE 802.11 (Wi-Fi), IEEE 802.15.1 (Bluetooth) та IEEE 802.15.4 (LR-WPAN) [93].

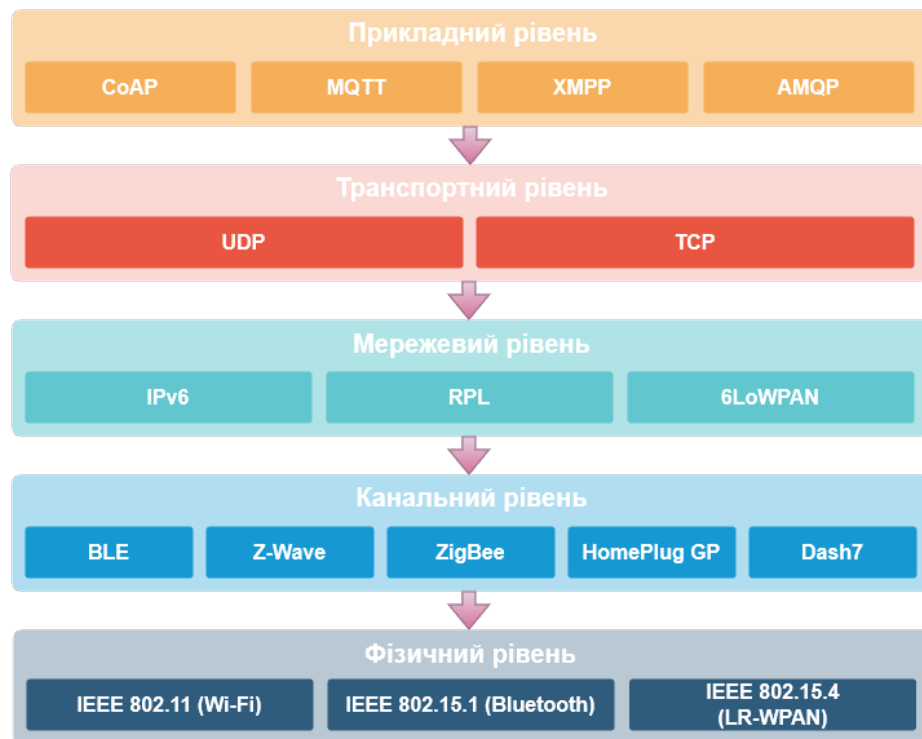


Рисунок 2.4 – Стек протоколів, що використовуються в системах Розумний будинок та IoT [93]

У контексті розробки онлайн ВС для ГЛ GOLDi, яке повинне забезпечити можливість здобути практичні навички з розробки апаратного та програмного забезпечення для роботи з системами Розумний будинок, потрібно проаналізувати протоколи передачі даних прикладного рівня, оскільки саме з ними взаємодіють ПАК та користувачі.

2.5.2 CoAP

CoAP (англ. – Constrained Application Protocol, укр. – протокол обмеженого застосунку) – це протокол прикладного рівня синхронного та асинхронного обміну даними за принципом запитів та відповідей. CoAP був розроблений для того, щоб малопотужні пристрої з низьким

енергоспоживанням, обчислювальними та комунікаційними можливостями могли використовувати принципи REST-взаємодії (англ. – Representational State Transfer, укр. – передача репрезентативного стану). CoAP – це протокол вебпередачі, подібний до HTTP, здатний забезпечити REST-архітектуру при використанні протоколу 6LoWPAN. Очікується, що після завершення специфікації CoAP у майбутньому в різних сферах застосування IoT будуть розгорнуті мільйони пристроїв, що використовуватимуть цей протокол. Даний протокол можливо використовувати для інтелектуальної енергетики, інтелектуальних електромереж, керування будинками, інтелектуального керування освітленням, промислових систем керування тощо [94].

CoAP – це двійковий протокол, який працює поверх протоколу UDP. Підрівень обміну повідомленнями додає тонкий рівень керування, який забезпечує виявлення дублікатів і, якщо треба, надійну доставку повідомлень на основі простого механізму зупинки та очікування повторних передач. Причина розробки протоколу прикладного рівня на основі UDP для керування ресурсами полягає в бажанні відмовитися від накладних витрат протоколу TCP та знизити вимоги до пропускну здатності мережі. Незважаючи на те, що CoAP був створений для IoT, він не забезпечує жодних вбудованих функцій безпеки [93].

Отже, CoAP – це протокол, який працює в реальному часі та забезпечує синхронний / асинхронний обмін даними за принципом запит / відповідь [93].

2.5.3 MQTT

MQTT (англ. – Message Queue Telemetry Transport, укр. – передавач черги повідомлень телеметрії) – це асинхронний протокол обміну даними за архітектурою публікації / підписки, який працює поверх протоколу TCP. MQTT був розроблений компанією IBM і призначений для полегшення комунікації між різними пристроями. MQTT розроблений для з'єднання вбудованих пристроїв і мереж із застосунками та проміжним ПЗ. MQTT

використовує архітектуру публікації / підписки для забезпечення масштабованості системи та простоти її реалізації. Протокол MQTT може забезпечувати маршрутизацію для малопотужних пристроїв з малим об'ємом пам'яті в мережах з низькою пропускнуою здатністю. У MQTT є брокер (сервер), який зберігає топіки (теми публікацій). Кожен ПАК може публікувати інформацію у певний топік, тобто відправляти інформацію брокеру на певну тему, та / або отримувати повідомлення автоматично кожного разу, коли з'являється нове повідомлення в темі, на яку підписаний ПАК [94].

MQTT був розроблений спеціально для застосунків телеметрії з трьома конкретними цілями проектування:

- він повинен пропонувати одноразовий гарантований режим доставки, що дозволяє надійно передавати повідомлення на всьому шляху від віддаленого датчика до пристрою;
- протокол має бути якомога легшим, оскільки більшість віддаленої телеметрії виконується в мережах з низькою пропускнуою здатністю, тому мінімізація накладних витрат на кожне повідомлення дуже важлива;
- протокол повинен бути дуже простим для реалізації на вбудованих пристроях, таких, як датчики [94].

Пристрої можуть змінюватися від датчиків та виконавчих механізмів до смартфонів, вбудованих систем на транспортних засобах, ноутбуків та повнофункціональних комп'ютерів [94].

Отже, MQTT – це протокол передачі телеметрії, який працює в реальному часі та забезпечує асинхронний обмін даними за принципом публікації / підписки [94].

2.5.4 XMPP

XMPP (англ. – Extensible Messaging and Presence Protocol, укр. – розширюваний протокол обміну повідомленнями та інформацією про

присутність) був розроблений для обміну повідомленнями та месенджерів. Він був стандартизований IETF. XMPP добре зарекомендував себе, тому часто використовується в різних Інтернет-системах. Проте великі компанії швидко відмовилися від нього, оскільки тривалий час протокол не оновлювався. XMPP підтримує архітектуру публікації / підписки, яка більше підходить для IoT, але також використовує і підхід запит / відповідь для отримання інформації про присутність. XMPP – це стандарт обміну миттєвими повідомленнями IETF, який використовується для чатів, голосових та відеодзвінків. Багато функцій XMPP роблять його кращим протоколом для більшості програм обміну миттєвими повідомленнями та актуальним в межах IoT. Він децентралізовано працює на різних Інтернет-платформах. XMPP безпечний і дозволяє додавати нові програми поверх основних протоколів [94].

Отже, XMPP – це протокол обміну повідомленнями, який працює в реальному часі та забезпечує синхронний / асинхронний обмін даними за принципом запит / відповідь та публікації / підписки [93].

2.5.5 AMQP

AMQP (англ. – Advanced Message Queuing Protocol, укр. – розширений протокол черги повідомлень) – це асинхронний протокол, який виник у фінансовій сфері. Він може використовувати різні транспортні протоколи, але передбачає наявність надійного базового транспортного протоколу, наприклад, TCP. AMQP забезпечує асинхронну архітектуру публікації / підписки для комунікації між ПАК. Його головною перевагою є функція проміжного зберігання, що забезпечує надійність навіть після збоїв у роботі мережі. AMQP підтримує надійний зв'язок за допомогою примітивів гарантії доставки повідомлень, включно з доставкою не більше одного разу, не менше одного разу та лише один раз. AMQP визначає рівень обміну повідомленнями поверх свого транспортного рівня [94].

Отже, AMQP – це протокол, який використовується у фінансовій сфері та забезпечує асинхронний обмін даними за принципом публікації / підписки [94].

2.5.6 Порівняння протоколів передачі даних прикладного рівня, що використовуються в системах Розумний будинок та IoT

Порівняння протоколів передачі даних прикладного рівня, що використовуються в системах Розумний будинок та IoT представлено у вигляді таблиці 2.2 [93].

Отже, проведені у наукових працях [93]-[94] дослідження показали, що протокол передачі даних MQTT найкраще підходить для використання в системах Розумний будинок та IoT, оскільки він здатний забезпечити надійну та безпечну маршрутизацію для невеликих та дешевих пристроїв з низьким енергоспоживанням і низьким рівнем пам'яті у нестабільних мережах та у мережах з низькою пропускнуою здатністю.

Таблиця 2.2 – Порівняння протоколів передачі даних прикладного рівня, що використовуються в системах Розумний будинок та IoT [93]

Критерій \ Протокол	CoAP	MQTT	XMPP	AMQP
1	2	3	4	5
Мета	Зменшити накладні витрати на повідомлення, їх розмір та необхідність їх фрагментації	Забезпечити надійну комунікацію в мережах з обмеженою пропускнуою здатністю чи нестабільним зв'язком	Забезпечити користувачам можливість підключатися до кількох систем обміну миттєвими повідомленнями	Для передачі ділових повідомлень між ПАК
Режим обміну повідомленнями	Синхронний та асинхронний	Асинхронний	Синхронний та асинхронний	Асинхронний
Працює в реальному часі	+	+	+	—

Продовження таблиці 2.2

1	2	3	4	5
Надійність	Вбудований механізм забезпечення надійності обміну повідомленнями	Надійність доставки повідомлень через різні рівні QoS	TCP гарантує надійність	Функція проміжного зберігання забезпечує надійність
Безпека	DTLS	SSL / TLS	SSL / TLS	SSL / TLS
Підтримка опції «якість обслуговування» (QoS)	+	+	–	+
Архітектура	Запит / відповідь	Публікація / підписка	Публікація / підписка, запит / відповідь	Публікація / підписка
Протокол транспортного рівня	UDP	TCP	TCP	TCP
Споживання енергії	Зменшує споживання енергії	Зменшує споживання енергії	Зменшує споживання енергії	Споживання більше, ніж у MQTT
Пропускна здатність мережі	Збільшує пропускну здатність мережі	Оптимізує пропускну здатність мережі	Сильно зменшує пропускну здатність мережі	Зменшує пропускну здатність мережі

2.6 Висновки до розділу 2

У даному розділі було проведено дослідження особливостей існуючих ВС. У результаті було запропоновано класифікацію ВС на основі двох критеріїв – технологія реалізації та сфера застосування. Розроблена класифікація дозволила прийняти рішення щодо реалізації розроблюваної ВС як неімерсійної системи у вигляді вебклієнту.

Крім цього, було досліджено методи організації взаємодії користувачів з ВС. Аналіз показав, що у ВС можуть використовуватися різні види UI. У ВС використовується багато методів організації взаємодії на основі GUI. Можна виділити WIMP-методи та специфічні для ВС методи. На основі проведеного аналізу було запропоновано класифікацію методів організації взаємодії користувачів з ВС.

У даному розділі отримав подальший розвиток метод організації взаємодії користувачів з ВС завдяки впровадженню методу віддаленого програмування для надання нових засобів отримання даних, підвищення зручності їх обробки та наочності подання.

Для реалізації онлайн ВС для ГЛ GOLDi було обрано GE Unreal Engine 4, оскільки дана система пропонує високий рівень графічних можливостей, готові шаблони ігор та моделей, текстур та матеріалів для них. Важливим є те, що за допомогою даного GE можливо створювати ігри для веббраузерів, тобто HTML5-сумісні програми [80].

Для реалізації методу віддаленого програмування було обрано протокол передачі даних прикладного рівня MQTT для забезпечення взаємодії між ВС і ПАК користувачів, оскільки це популярний та надійний протокол передачі даних, який часто використовується в системах Розумний будинок та IoT [94].

З ПРОЄКТУВАННЯ ОНЛАЙН ВІРТУАЛЬНОГО СЕРЕДОВИЩА РОЗУМНИЙ БУДИНОК ДЛЯ ГІБРИДНОЇ ЛАБОРАТОРІЇ GOLDI

3.1 Аналіз вимог до віртуального середовища

ВС призначене для використання в навчальному процесі для навчання створенню програмного та апаратного забезпечення для взаємодії з системами «розумного» будинку та для проведення навчальних експериментів для перевірки розроблених сценаріїв автоматизації «розумного» будинку.

Можна виділити основні групи користувачів розроблюваного ВС:

- студенти, які використовують його для отримання: теоретичних знань з систем Розумний будинок та IoT, їх обладнання та технологій; практичних навичок розробки ПАК для роботи з такими системами, а також сценаріїв домашньої автоматизації;
- викладачі, які використовують його в процесі викладання курсів з систем домашньої автоматизації, Розумний будинок та IoT для створення практичних завдань для студентів з метою кращого засвоєння ними отриманих знань.

Навчальні завдання та досвід, який студенти можуть отримати в результаті їх вирішення, представлені у вигляді діаграми прецедентів, що наведена на рисунку 3.1. З діаграми видно, що отримання знань про архітектуру систем Розумний будинок та IoT також потребує вивчення видів обладнання для домашньої автоматизації, основ його функціонування та принципів розміщення у будинку. Для розробки власних ПАК для взаємодії з системами Розумний будинок та IoT студент повинен вивчити протокол передачі даних MQTT, а також вміти обирати інструментарій для їх розробки. Крім цього, студент може отримати досвід розробки сценаріїв домашньої автоматизації, що передбачає отримання досвіду у визначенні потреб мешканців таких будинків, а також симуляції їх поведінки.

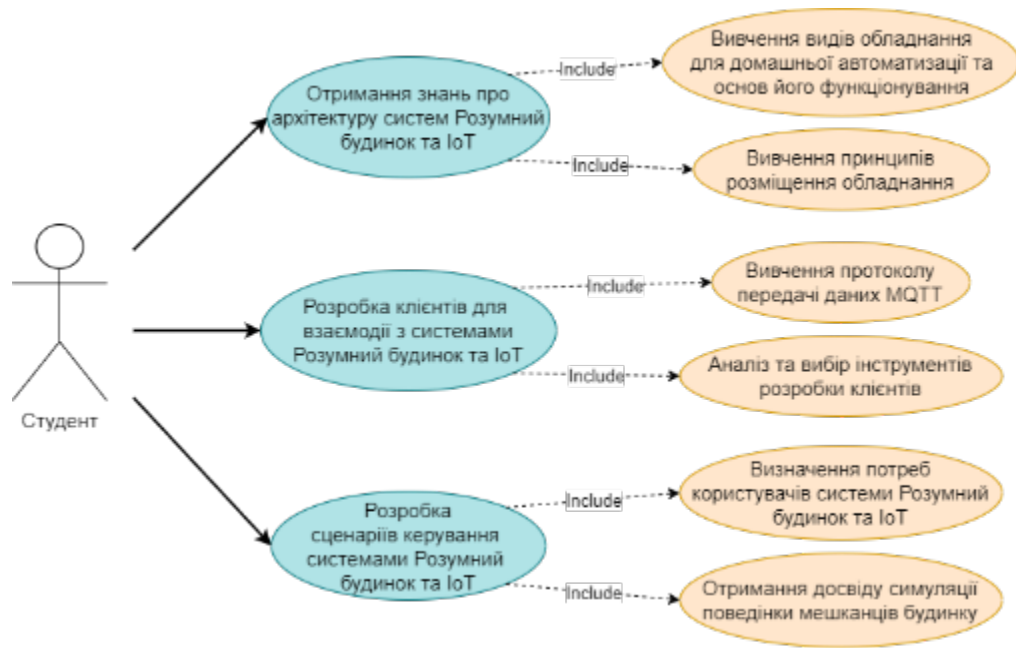


Рисунок 3.1 – Навчальні завдання студента

На основі аналізу виявлених груп користувачів можна виділити наступні вимоги до розроблюваного ВС за класифікацією вимог від компанії IBM [95].

Функціональні вимоги:

- симуляція будинку, який складається з кімнат. Кімнати – це прості блоки, у яких можна розмістити обладнання;
- симуляція обладнання, такого, як: вимикач, клімат-контролер, джерела світла, система клімат-контролю, датчики, двері, вікно;
- дистанційне керування обладнанням будинку через MQTT;
- переміщення будинком з можливістю взаємодії з обладнанням.

Результат взаємодії повинен відображатися у ВС і передаватися підключеним програмам / пристроям через MQTT;

- ведення та відображення на екрані журналу обміну повідомленнями між пристроями будинку та підключеними до ВС ПАК користувача;

- джерела світла повинні впливати на датчики освітленості, тобто датчики освітлення повинні реагувати на зміну освітленості;

- вимикач повинен вмикати / вимикати прив'язане джерело світла та надсилати / отримувати сповіщення про зміну статусу прив'язаного джерела світла (повідомлення про включення / вимкнення);
- система клімат-контролю повинна впливати на датчики температури та вологості, тобто ці датчики повинні реагувати на зміну відповідних параметрів ВС;
- клімат-контролер повинен:
 - мати можливість встановити значення температури у форматі з рухомою комою, а значення вологості у цілочисельному форматі;
 - вмикати або вимикати прив'язану систему клімат-контролю;
 - відображати попередньо встановлену температуру та вологість;
 - надсилати / отримувати сповіщення про зміну стану прив'язаної системи клімат-контролю (повідомлення про включення / вимкнення, зміну температури, вологості).

Нефункціональні вимоги:

- ВС повинна бути доступна онлайн за допомогою останніх версій веббраузерів Google Chrome (89.0.4389.90) і Mozilla Firefox (86.0.1);
- мова ВС – англійська;
- система повинна використовувати протокол передачі даних MQTT для зв'язку між ВС і зовнішніми програмами / пристроями користувачів;
- TCP/IP-сокети повинні використовуватися для підключення ПАК до ВС;
- користувач повинен мати можливість надсилати команди ВС за допомогою своїх ПАК. У результаті користувач повинен бачити зміну стану ВС чи помилку;
- ПАК повинні мати можливість отримувати повідомлення про зміну стану обладнання у будинку ВС;
- повинна бути забезпечена можливість підписатися на всі теми, що стосуються обладнання у будинку ВС;

- для підключення та подальшого зв'язку між екземпляром ВС та ПАК користувача має використовуватися ідентифікатор сеансу (IS), який разом з іншими параметрами (хостом та портом) для під'єднання ПАК повинен відображатися користувачу на екрані;
- IS повинен залишатися «живим» протягом однієї години;
- журнал обміну повідомленнями повинен складатися з журналу всіх подій (оновлення статусу з'єднання та отримані повідомлення) та журналу відправлених «розумним» будинком повідомлень;
- кожна кімната повинна мати власний ідентифікатор, який повинен бути однаковим між сеансами роботи ВС. Ідентифікатор кімнати повинен відображатися на підлозі цієї кімнати;
- кожне обладнання повинне мати власний ідентифікатор, який повинен бути однаковим між сеансами роботи ВС. Підказка з ідентифікатором обладнання та можливим описом повинні відображатися над обладнанням;
- використання перспективи зверху вниз для спостереження за ігровим персонажем;
- використання комп'ютерної миші для вільного переміщення віртуальним світом, тобто агент рухається в напрямку місця, на яке користувач клацнув лівою кнопкою миші;
- використання комп'ютерної миші та клавіатури для взаємодії з об'єктами ВС. За допомогою миші користувач повинен обрати об'єкт для взаємодії та натиснути клавішу F на клавіатурі і, якщо ігровий персонаж знаходиться близько до нього, елемент повинен виконати певну дію чи відобразити віджет для подальшої взаємодії;
- підсвічування активних елементів віртуального світу;
- відображення спеціальної позначки над об'єктами, з якими може взаємодіяти агент, коли він потрапляє у приміщення будинку;
- система клімат-контролю повинна бути керованою за допомогою клімат-контролеру та дистанційно через MQTT;

- клімат-контролер повинен мати можливість бути прив'язаним до зазначеної системи клімат-контролю в приміщенні. Параметри прив'язки можна змінити лише в налаштуваннях проєкту;
- вимикач повинен мати два стани – увімкнено та вимкнено та мати можливість бути прив'язаним до певного джерела світла в кімнаті. Параметри прив'язки можна змінити лише в налаштуваннях проєкту;
- джерела світла повинні бути керовані дистанційно через MQTT та за допомогою вимикача. Джерела світла можуть бути двох типів – звичайні та RGB. Колір світла RGB-джерел можна встановити лише дистанційно через MQTT. Колір повинен бути представлений у форматі HSV (Hue (відтінок), Saturation (насиченість), Value (Brightness, яскравість)). RGB-джерело може бути увімкнено за допомогою вимикача із кольором, який раніше був встановлений дистанційно через MQTT;
- датчики можуть бути чотирьох типів: освітленості, температури, вологості та руху. Вони повинні надсилати вимірювання у міжнародній системі одиниць із зазначеним у налаштуваннях проєкту періодом.

Процес взаємодії користувача з розроблюваним ВС представлений у вигляді діаграми прецедентів, що наведена на рисунку 3.2.



Рисунок 3.2 – Діаграма прецедентів для процесу взаємодії користувача з розроблюваним віртуальним середовищем

З діаграми видно, що проведення експерименту передбачає виконання низки інших дій, таких, як: ініціалізація сеансу, під'єднання ПАК, переміщення у ВС, взаємодія з обладнанням, дистанційне керування обладнанням, перегляд журналу подій тощо. Для того, щоб мати можливість дистанційного керування, користувач повинен під'єднати свої ПАК до ВС. Після цього він матиме можливість дистанційної взаємодії з обладнанням ВС.

Отже, виходячи з аналізу вимог, можна зробити висновок про те, що розроблюване онлайн ВС є комплексним ПЗ, оскільки повинне надавати як реалістичний віртуальний тривимірний «розумний» будинок, так і API для забезпечення можливості дистанційного керування його обладнанням. Крім цього, слід взяти до уваги те, що у майбутньому функціонал ВС може бути розширений. Усі ці фактори вимагають реалізації масштабованої та гнучкої архітектури системи.

3.2 Розробка архітектури програмної системи

У процесі аналізу вимог до розроблюваного ПЗ було вирішено реалізувати систему у вигляді клієнт-серверної архітектури, тому функціонал застосунку було вирішено розділити на два окремих програмних продукти (ПП). Загальну схему архітектури розроблюваної системи представлено на рисунку 3.3.

Сервер здійснює підключення ПАК користувача до його екземпляра ВС. Він веде облік сеансів роботи користувачів та надає REST API для ВС, за допомогою якого ВС може ініціалізувати дії щодо сеансу роботи, такі, як початок нового сеансу, продовження та завершення сеансу тощо. Для обладнання ВС та ПАК користувача сервер надає MQTT-брокер, який виступає маршрутизатором повідомлень. Сервер повинен генерувати ІС та надсилати його разом з іншими параметрами підключення (наприклад, хост та порт) у відповідь на запит про початок нового сеансу від об'єкту ВС. За допомогою даного ідентифікатора сервер повинен асоціювати об'єкт ВС з

його MQTT-брокером. Інформація щодо активних сеансів роботи буде зберігатися в базі даних (БД) SQLite, оскільки вона реалізує автономний, безсерверний (тобто забезпечує локальне зберігання даних), транзакційний механізм баз даних SQL без необхідності додаткового конфігурування, а тому добре підходить для використання у малих серверах [96].

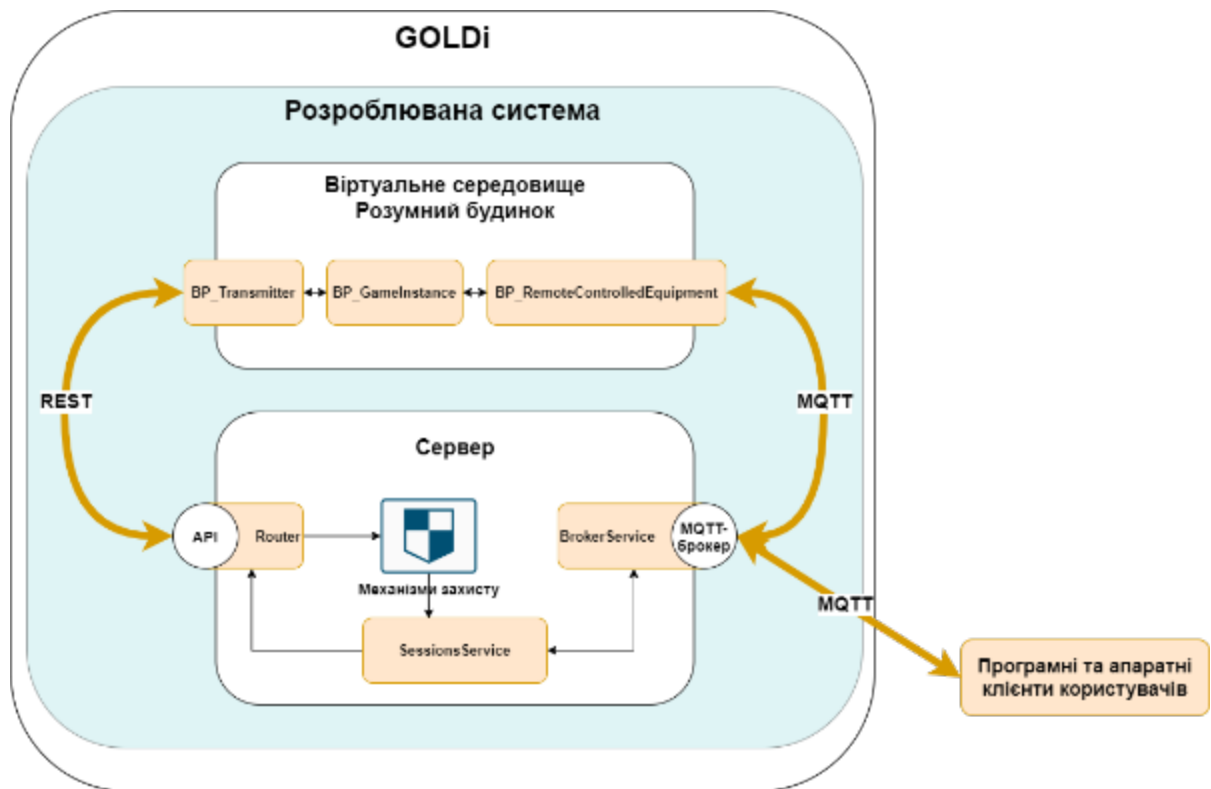


Рисунок 3.3 – Архітектура програмної системи

ВС Розумний будинок повинне бути реалізоване як неімерсійна система у вигляді вебклієнту для запуску у веббраузерах Google Chrome і Mozilla Firefox. Даний ПП повинен виконувати усі функції симуляції ВС та його обладнання, а також надавати можливості для взаємодії користувача з ним. Дане ВС повинне ініціювати дії стосовно сеансу роботи, надсилаючи запити про це серверу за допомогою його REST API. Воно повинне відображати параметри для під'єднання ПАК користувача, журнал обміну повідомленнями та іншу інформацію користувачу.

На рисунку 3.4 представлено діаграму класів серверу.

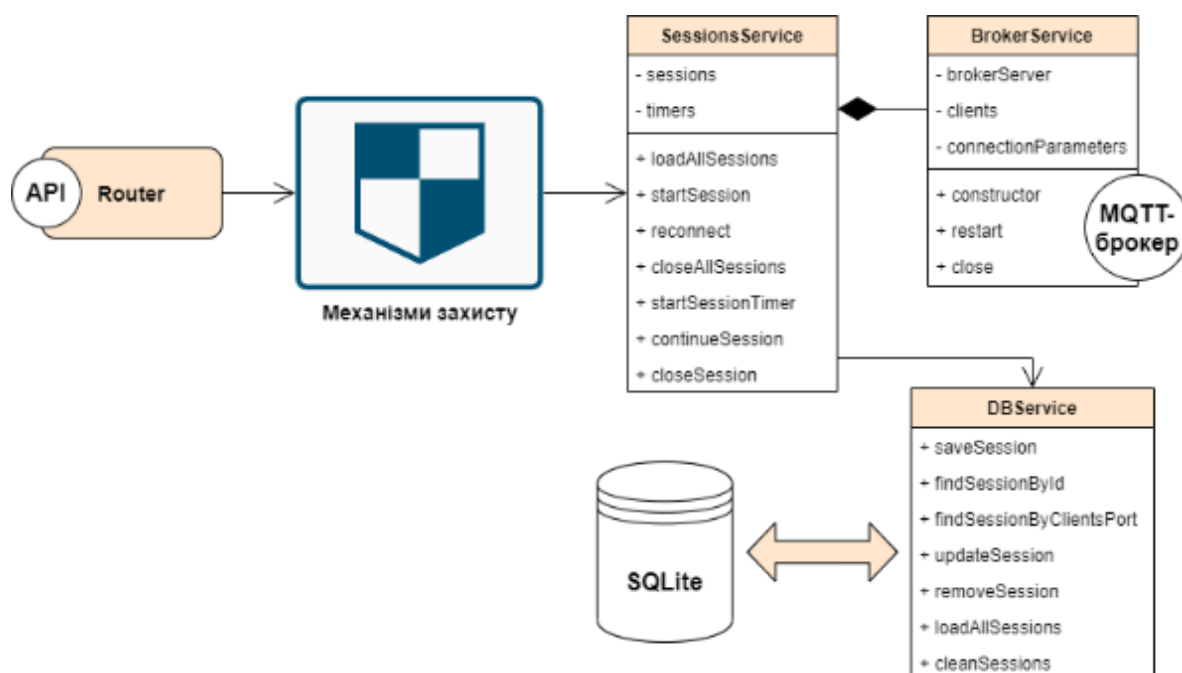


Рисунок 3.4 – Діаграма класів серверу

Точкою входу для ВС є API серверу, за який відповідає модуль Router. Даний модуль отримує запити та перевіряє їх на безпеку, після чого інтерпретує та викликає відповідні функції класу SessionsService. Отриманий результат Router повертає ВС як відповідь на його запит. До механізмів захисту належать обмежувач кількості запитів з одного IP за певний період часу та захист заголовків HTTP-відповідей, який прибирає небезпечні заголовки. Основним класом є SessionsService, який відповідає за ведення обліку активних сеансів роботи користувачів. Даний модуль спроектований за архітектурним шаблоном Одинак. Він генерує IC та зберігає за цими ідентифікаторами об'єкти класу BrokerService у словнику sessions. Крім цього, SessionsService запускає таймери сеансів на одну годину та зберігає їх за ідентифікатором у словнику timers. Проте час сеансу може бути подовжено, якщо ВС надішле відповідний запит. До основних методів даного класу належать loadAllSessions, startSession, reconnect, continueSession тощо. SessionsService використовує клас DBService, який також реалізований за шаблоном Одинак. Даний модуль надає методи для збереження, отримання та пошуку даних у БД. Як вже було сказано, для кожного сеансу SessionsService

створює об'єкт класу BrokerService, який відповідає за запуск MQTT-брокеру для сеансу, облік підключених до нього ПАК, а також за збереження параметрів підключення ПАК до брокеру.

На рисунку 3.5 представлено діаграму класів ВС Розумний будинок.

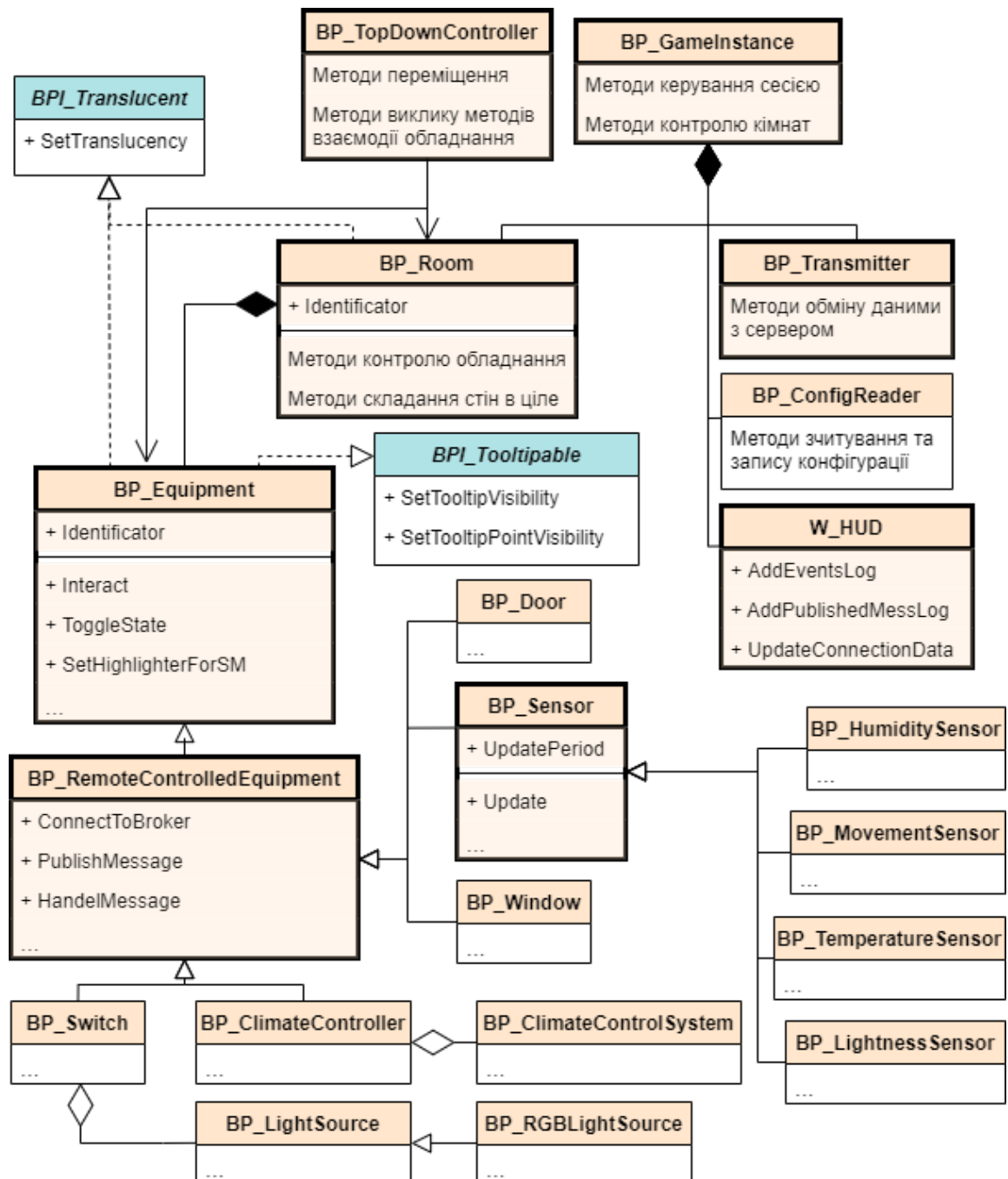


Рисунок 3.5 – Діаграма класів віртуального середовища Розумний будинок

Одним з найголовніших класів є клас **BP_GameInstance**, який виконує відображення та оновлення HUD за допомогою класу **W_HUD**, керування

сеансом (запуск таймерів, зчитування конфігурації з файлу config.ini для під'єднання до API серверу за допомогою класу BP_ConfigReader, надсилання запитів щодо сеансу на сервер за допомогою класу BP_Transmitter), збереження та контроль кімнат для того, щоб надавати команди для початку під'єднання обладнання кімнат до MQTT-брокеру. Клас W_HUD містить інтерфейс HUD та надає методи для його оновлення, наприклад, додати новий запис у журнал подій чи відобразити нові параметри підключення ПАК. Клас BP_Room відображає кімнату, яка складається з полу та блоків стін (цільна стіна, стіна з прорізом для вікна, стіна з прорізом для дверей) та виконує розрахунки для складання стін з перелічених стінних блоків. Крім цього, BP_Room зберігає обладнання, яке розташоване у межах даної кімнати, та передає йому команди щодо комунікації з брокером від BP_GameInstance, а також може викликати інші методи обладнання по команді від BP_TopDownController (наприклад, команда стати прозорим). Коли користувач потрапляє у кімнату, вона викликає метод для відображення позначки над обладнанням, з яким може взаємодіяти агент.

Клас BP_Equipment – це абстрактний базовий клас для всього обладнання. Він (так само, як і BP_Room) реалізує інтерфейс BPI_Translucent, який потрібен для перемикавання режиму відображення об'єкту – прозорий / непрозорий. Крім цього, BP_Equipment реалізує інтерфейс BPI_Tooltipable для відображення підказки над обладнанням. BP_Equipment надає методи для перемикавання свого стану та взаємодії з собою, які можуть бути перевизначені у класах-спадкоємцях, а також надає інші службові методи. Класи BP_Equipment та BP_Room мають поле Identifier. Абстрактний клас BP_RemoteControlledEquipment наслідує клас BP_Equipment та реалізує функціонал для комунікації з MQTT-брокером. Він має методи для під'єднання до брокеру, публікації, отримання та обробки повідомлень тощо. Даний клас успадковується усіма класами обладнання, для якого передбачено дистанційне керування за допомогою MQTT: BP_Switch, BP_ClimateController, BP_Door, BP_Window та BP_Sensor. Класи

BP_ClimateControlSystem та BP_LightSource (BP_RGBLightSource, який є нащадком BP_LightSource) не є підключеним до брокеру обладнання, а контролюються через клас-контролер, відповідно BP_ClimateController та BP_Switch. BP_ClimateControlSystem та BP_LightSource виконують симуляцію вигляду та поведінки обладнання, а BP_ClimateController та BP_Switch надають можливість дистанційного керування ними, а також віджети-інтерфейси для керування ними користувачем за допомогою GUI.

Клас BP_Sensor є нащадком класу BP_RemoteControlledEquipment та є абстрактним базовим класом для всіх датчиків. Він має поле UpdatePeriod, за допомогою якого можна вказати інтервал, з яким датчик буде надсилати нові вимірювання. Метод Update потрібен для розрахунку нових значень вимірювань. Даний клас успадковується класами BP_HumiditySensor, BP_MovementSensor, BP_TemperatureSensor та BP_LightnessSensor.

BP_TopDownController – це контролер ігрового персонажу. Даний клас відповідає за переміщення агента та його взаємодію з кімнатами та обладнанням. Коли агент входить у кімнату і між ігровою камерою та агентом стає стіна, даний клас викликає метод SetTranslucency класу BP_Room, який так само викликає цей метод в об'єктів класу BP_Equipment або його нащадків, які знаходяться у даній кімнаті. Крім цього, якщо користувач клацнув на певному обладнанні у ВС та натиснув F для взаємодії з ним, даний клас розраховує відстань від агента до активного пристрою і, якщо він поряд, викликає метод Interact даного пристрою.

Отже, архітектура ПЗ побудована на основі клієнт-серверної архітектури та складається з серверу та ВС. Це дозволяє розмежувати функціонал на два окремих ПП. Розроблені архітектури серверу та ВС є гнучкими та масштабованими, тому реалізація нового функціоналу в майбутньому не потребуватиме значних змін у вже реалізованому функціоналі. Для забезпечення можливості комунікації між сервером та ВС, а також між сервером та ПАК потрібно розробити архітектуру API серверу і ВС.

3.3 Розробка архітектури прикладного програмного інтерфейсу

3.3.1 Точки входу серверу

Як вже було зазначено у підрозділі 3.2, для комунікації між сервером та ВС буде використовуватися REST API. REST API – це API, який використовує архітектурний стиль передачі репрезентативного стану (REST). Архітектурний стиль REST використовує HTTP для запиту доступу та використання даних [97].

REST працює поверх протоколу HTTP. Він використовує стандартні можливості HTTP, такі, як методи GET, PUT, POST і DELETE. Коли запит надсилається до REST API, відповідь (подання цільового інформаційного ресурсу) повертається у форматі JSON, XML або HTML. REST API визначається вебадресою або уніфікованим ідентифікатором ресурсу (URI), який зазвичай дотримується певної конвенції про іменування запитів для роботи з REST API сервером [97].

Точкою входу (англ. – endpoint) серверу вважається вебадреса серверу, на яку можна відправити запит та отримати коректне подання бажаних даних. Перелік точок входу серверу для комунікації з ВС наведено у таблиці 3.1 [97].

Строкове поле `session_id` (IC) використовується усіма запитами, крім запиту `/session`, для ідентифікації сеансу, до якого належить ВС та щодо якого він виконує дії. Вхідний параметр `session_id` передається на сервер як частина URI-строки. Сервер повертає результати у відповідь на запит у форматі JSON як JSON-об'єкт. Якщо у результаті обробки запиту виникла помилка, тоді буде повернено JSON-об'єкт з текстовим полем `error_message`, у якому буде зберігатися повідомлення про помилку [98].

Запит `/session` повертає `session_id`, а також хост (host) та порт (clientsPort) для підключення обладнання ВС та ПАК до MQTT-брокеру. Поле `timeLeft` визначає тривалість сеансу у мілісекундах, а поле `timeToDecide` визначає додатковий час у мілісекундах, протягом якого ВС повинен запропонувати користувачу продовжити або завершити сеанс, а після його

завершення надіслати запит для завершення сеансу. Поля `isContinued`, `isFinished` та `isStarted` зберігають результат виконання відповідного запиту.

Сервер у відповідь може повернути наступні коди стану HTTP: 200 (усе добре), 400 (неправильний `session_id`), 404 (звернення до неіснуючої точки входу), 429 (забагато запитів), 500 (сталася помилка під час обробки запиту).

Отже, розроблений перелік точок входу серверу для комунікації з ВС надає ВС багато функцій для керування сеансом.

Таблиця 3.1 – Точки входу серверу

Точка входу	Метод	Вхідні параметри	Відповідь	Призначення
/session	GET	—	host: string; clientsPort: number; session_id: string; timeLeft: number; timeToDecide: number	Розпочати новий сеанс
/session/continue_session/session_id	POST	session_id	session_id: string; isContinued: boolean; timeLeft: number	Подовжити час сеансу
/session/finish_session/session_id	POST	session_id	session_id: string; isFinished: boolean	Завершити сеанс
/session/reconnect/session_id	POST	session_id	host: string; clientsPort: number; session_id: string; timeLeft: number; timeToDecide: number	Поновити з'єднання з сервером. ВС може використовувати цей запит у разі виникнення проблем з Інтернет-з'єднанням
/session/start_session_timer/session_id	POST	session_id	session_id: string; isStarted: boolean	Запустити таймер сеансу. ВС використовує цей запит для запуску таймеру на сервері після того, як усе обладнання було підключено до MQTT-брокеру

3.3.2 Структура MQTT-тем обладнання віртуального середовища

ВС повинне дотримуватися наступних правил іменування MQTT-тем обладнання та публікації в них:

- MQTT-теми чутливі до регістру;

- MQTT-теми структуровані за ієрархією, подібною до папок і файлів у файловій системі з використанням косої риски (/) як роздільника;
- у назвах тем використовується лише UTF-8;
- \$SYS є зарезервованою темою і використовується для публікації інформації про брокер;
- # (хеш-символ) – багаторівневий символ підстановки;
- клієнт може публікувати лише в окремій темі. Тобто використання знаків підстановки під час публікації заборонено. Наприклад, щоб опублікувати повідомлення у дві теми, клієнту необхідно опублікувати повідомлення окремо у кожен з них [99].

Структура MQTT-тем обладнання ВС представлена в таблиці 3.2.

Таблиця 3.2 – Структура MQTT-тем обладнання віртуального середовища

Тема	Структура
Вітальна тема	ідентифікатор_сеансу/hello
\$SYS-тема	ідентифікатор_сеансу/\$SYS/...
Обладнання	ідентифікатор_сеансу/ідентифікатор_кімнати/ тип_обладнання/ідентифікатор_обладнання/...
Підписка на усі несистемні теми	ідентифікатор_сеансу/#
Підписка на усі теми обладнання у кімнаті	ідентифікатор_сеансу/ідентифікатор_кімнати/#
Підписка на усі теми конкретного типу обладнання у кімнаті	ідентифікатор_сеансу/ідентифікатор_кімнати/ тип_обладнання/#

Отже, розроблена структура MQTT-тем є доволі простою та легкою для використання. Для розробки системи за запропонованою архітектурою системи та API серверу потрібно вибрати інструментарій для розробки серверу, оскільки для реалізації ВС у підрозділі 2.4 вже було обрано Unreal Engine 4.

3.4 Обґрунтування вибору інструментарію розробки

3.4.1 Вибір стеку технологій розробки

Для розробки серверу потрібно обрати МП, яка використовується у backend-розробці. Backend – це технічний термін, який використовується комп’ютерними фахівцями та програмістами та узагальнює код програміста, який виконує операції на стороні сервера, включно з функціями CRUD (англ. – create, read, update, delete, укр. – створити, отримати, змінити, видалити), функціями роботи з БД і всією логікою сервера. Крім цього, важливо обрати фреймворк для розробки, оскільки використання фреймворків спрощує розробку застосунків і пришвидшує час їх розробки [100].

Найпопулярнішими мовами програмування у backend-розробці є JavaScript, Python, PHP та Ruby [100].

Протягом останніх восьми років JavaScript займає першу позицію як найчастіше використовувана технологія програмування в щорічних опитуваннях Stack Overflow. JavaScript відомий завдяки простоті та високим темпам розробки застосунків. TypeScript – це типізована версія JavaScript [100].

За допомогою платформи Node.js можливо виконувати операції на стороні сервера, використовуючи JavaScript. Але також важливо розуміти, що Node.js працює як платформа і здебільшого використовує фреймворк Express.js. Express.js – це серверний фреймворк з відкритим вихідним кодом для Node.js. Express.js пропонує модулі проміжного ПЗ для вирішення різних проблем розробки. Такі модулі можуть надавати логери HTTP-запитів, обробники помилок, захист HTTP-заголовків тощо [100]-[101].

ПЗ, розроблене з використанням Node.js, працює в одному процесі, без створення нового потоку для кожного запиту, що робить розробку програми ефективнішою. Node.js демонструє високі показники швидкості, ефективності та гнучкості навіть у програмах із великим обсягом даних, які повинні працювати на різних пристроях [100]-[101].

Python. Python був створений в 1991 році. Це об'єктно-орієнтована високорівнева МП загального призначення, яка стала дуже популярна у веброзробці. Її відмінність полягає в тому, що вона більш консервативна і, як правило, існує лише один спосіб вирішити проблему з Python. Дана МП вимоглива і зрозуміла, але також відносно проста для вивчення, що може бути причиною того, що Python дуже популярний у наукових спільнотах і викладається в університетах. Python є універсальним і його широко використовують і вибирають технічні гіганти, такі, як NASA, Google, Netflix, Spotify, Instagram, Reddit тощо [100]-[101].

Django – один з найпопулярніших фреймворків Python з відкритим кодом. Завдяки принципу DRY (англ. – Don't Repeat Yourself, укр. – не повторюйся) Django повторно використовує код, який вже є, і фокусується на нових частинах, що робить роботу розробника неймовірно швидкою. Ще одна перевага Django – безпека. Це, серед іншого, є причиною того, що його часто вибирають для розробки ПП у сфері фінансів, наук про дані, систем бронювання тощо. Для нього було створено багато різних бібліотек, які готові до використання в різних середовищах і проєктах [101].

PHP. PHP є популярною серверною скриптовою МП. PHP має відкритий початковий код і зазвичай використовується для вебсайтів. Згідно з недавнім опитуванням W3Techs близько 79,1% вебсайтів в Інтернеті використовують PHP на сервері. PHP проста у використанні МП та добре працює з БД. Наявність сучасних фреймворків, великої спільноти, надійної кодової бази та простота розгортання робить цінною дану технологію [100].

Laravel – це один з найкращих безкоштовних фреймворків PHP, який було створено в 2011 році. Laravel відомий завдяки простоті розгортання та простому синтаксису. Для створення програм зі стабільною інфраструктурою цей фреймворк надає вбудований механізм шаблонів Blade. Крім цього, Laravel у комплекті має вбудовані функції тестування [102].

Ruby. Ruby – це backend-технологія з відкритим кодом, розроблена в 1990-х роках. Ця скриптова мова має схожість з Python, Java та Perl. Ruby

зазвичай використовується для розробки вебзастосунків і вважається фаворитом для створення застосунків-прототипів. Технологічні компанії, такі, як Esty, Airbnb і Shopify, використовують МП Ruby як технологію створення скриптів завдяки її об'єктно-орієнтованості, масштабованості та гнучкості. Завдяки простому коду та доступності сторонніх бібліотек Ruby є дуже функціональною технологією. Ruby також підтримує метапрограмування. У метапрограмуванні розробник пише код на Ruby, а програма сама представляє інший код. Завдяки гнучким інструментам, таким, як примарні методи, інтроспекція та макроси, можливо швидко створювати скрипти [100].

Ruby on Rails – це фреймворк з відкритим початковим кодом для розробки вебзастосунків мовою програмування Ruby. Даний фреймворк поширюється за ліцензією MIT. Ruby on Rails побудований на архітектурі Модель-Вид-Контролер. Він надає можливості для простої інтеграції з різними БД, а тому часто використовується для розробки вебсайтів, систем керування сайтами, торгівельних платформ тощо [101].

Порівняння технологій для backend-розробки наведено в таблиці 3.3.

Таблиця 3.3 – Порівняння технологій для backend-розробки

Критерій \ Технологія	Node.js / Express.js	Django	Laravel	Ruby on Rails
1	2	3	4	5
Тип	Платформа / Фреймворк	Фреймворк	Фреймворк	Фреймворк
Мова програмування	JavaScript, TypeScript	Python	PHP	Ruby
Рік першого релізу	2010	2005	2011	2004
Архітектура	MVC	MVC	MVC	MVC
Легкість вивчення	Дуже простий	Простий	Простий	Простий
Орієнтований на застосунки	Для невеликих та простих застосунків	Для розробки різних за розміром захищених систем у науці, фінансах тощо	Для малих та середніх проєктів	Для складних систем типу систем керування сайтами

Продовження таблиці 3.3

1	2	3	4	5
Швидка розробка прототипу	Швидко можна створити працюючий прототип застосунку	Часто використовується для створення прототипів	Рідко використовується для розробки прототипів	Часто використовується саме для створення прототипів
Проміжне ПЗ	У комплекті має багато проміжного ПЗ: логери, захист HTTP-заголовків тощо	Надає можливості для його інтеграції	Можна додавати проміжне ПЗ для обробки вхідних запитів	Мала кількість бібліотек проміжного ПЗ, які додатково потрібно встановлювати
Засоби тестування	Потрібно додатково встановлювати	Вбудовані	Вбудовані	Вбудовані
Тип ліцензії	MIT	3-clause BSD License	MIT	MIT

Отже, у результаті проведеного аналізу було вирішено обрати платформу Node.js та фреймворк Express.js, які використовують МП JavaScript та TypeScript. TypeScript буде використано в якості основної МП через його типізованість, завдяки чому код буде ефективнішим, зрозумілішим, безпечнішим та передбачуваним. Node.js активно використовується для backend-розробки, а фреймворк Express.js дозволяє швидко створювати прості ПП, надаючи необхідне проміжне ПЗ для логування, обробки помилок та захисту заголовків HTTP-відповідей на запити. Для пришвидшення процесу розробки серверу також слід дослідити IDE, які використовуються серед Node.js-розробників [100]-[101].

3.4.2 Вибір інтегрованого середовища розробки

У результаті аналізу джерел [103]-[104] для порівняння було обрано наступні IDE для Node.js-розробки: IntelliJ IDEA, Visual Studio Code, Atom та Sublime Text 3.

IntelliJ IDEA. Цю IDE можна використовувати для розробки різними мовами програмування та з використанням різних фреймворків для них.

Наприклад, Node.js, Angular.js, HTML, CSS, JavaScript, Java, React тощо. IntelliJ вважається однією з найкращих платних IDE для розробки застосунків Node.js, оскільки має велику кількість вбудованих засобів розробки, інструментів для роботи з БД, систем контролю версій тощо. Ця IDE дозволяє Node.js-розробникам писати код, запускати його, налагоджувати та перевіряти результати [103]-[104].

Для розробки Node.js-застосунків потрібно буде встановити плагін для IntelliJ IDEA. Це активує ПЗ з кількома функціями, включно з виділенням синтаксису, підказками, автодоповненнями тощо. Його налагоджувач дозволяє додавати точки зупину процесу виконання, оцінювати вирази та робити багато іншого, щоб розробник міг швидко виправити свій код [104].

Visual Studio Code (VS Code). Microsoft Visual Studio Code підтримує Node.js і дозволяє без проблем компілювати ПЗ для цієї платформи. Це відоме безкоштовне IDE для Node.js-розробки з відкритим кодом, який розробники можуть використовувати для створення ПП й іншими мовами програмування. Крім цього, Visual Studio Code має вбудовані розширення та інструменти для JavaScript, а також безліч плагінів, які створила спільнота. VS Code дозволяє легко розробляти Node.js-проекти з використанням JavaScript і TypeScript [104].

Atom. Atom – це IDE на основі Electron (фреймворк від GitHub, який дозволяє створювати десктоп-застосунки за допомогою JavaScript), як і Visual Studio Code. Це IDE з відкритим початковим кодом, що має інтегрований інструментарій для розробки з використанням HTML, CSS, Node.js і JavaScript. У комплекті також є менеджер плагінів, який можна використовувати за допомогою командного рядка. Atom був популярним, коли він з'явився у 2014 році, проте у 2021 році він є повільнішим та складнішим у порівнянні з VS Code. Крім цього, підтримка різних МП гірша, ніж в інших редакторах, таких, як IntelliJ або VS Code [103]-[104].

Sublime Text 3. Sublime Text 3 – це пропрієтарний редактор коду. Використовуючи Sublime Text 3, можна легко перемикатися між проектами,

розділяти подання коду для його порівняння, налаштовувати редактор відповідно до потреб користувача тощо. Проте Sublime Text 3 містить менше функцій, ніж в інших IDE, наприклад, відсутній інтегрований термінал. Крім цього, процес встановлення плагінів зазвичай потребує пошуку плагінів у браузері, оскільки вбудований пошук гірший у порівнянні з іншими IDE [103]-[104].

Порівняння IDE для Node.js-розробки наведено в таблиці 3.4.

Таблиця 3.4 – Порівняння інтегрованих середовищ розробки для Node.js-розробки

Критерій \ IDE	IntelliJ IDEA	Visual Studio Code	Atom	Sublime Text 3
Тарифний план	Платне	Безкоштовне	Безкоштовне	Безкоштовне
Підтримка TypeScript	+	+	+	+
Підтримувані ОС	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows, Linux, MacOS
Наявність автодоповнення та виділення синтаксису	Після встановлення плагіну для Node.js	Наявне	Наявне	Наявне
Інструмент запуску тестів та відображення покриття тестами	+	+	Потрібно встановлювати плагін	Потрібно встановлювати плагін
Розширюваність плагінами	+	+	+	+
Швидкість запуску	Повільно запускається	Швидко запускається	Повільно запускається	Швидко запускається
Стабільність роботи	Стабільно працює	Стабільно працює	Можуть бути помилки після зміни конфігурації	Стабільно працює
Наявність розділеного перегляду файлу	+	+	+	+
Спільнота	Велика	Велика спільнота користувачів та розробників плагінів	Велика	Велика

Отже, у результаті проведеного порівняння IDE для Node.js-розробки було обрано Visual Studio Code, оскільки це безкоштовне IDE з вбудованими

плагінами підтримки розробки з використанням JavaScript, TypeScript та Node.js, інструментами тестування, великою кількістю інших плагінів та зручним інтерфейсом. Крім цього, дане IDE швидко запускається та стабільно працює [104].

3.5 Висновки до розділу 3

У даному розділі було сформульовано вимоги до розроблюваного онлайн ВС Розумний будинок для ГЛ GOLDi. У результаті проведеного аналізу вимог було розроблено архітектуру системи, яка побудована на основі клієнт-серверної архітектури та складається з серверу та ВС. Для серверу було розроблено структуру точок входу для комунікації між сервером та ВС, а для ВС було визначено правила іменування та структуру MQTT-тем обладнання та правила публікації в них.

Крім цього, додатково до обраного у розділі 2 інструментарію розробки ВС було обрано платформу Node.js, фреймворк Express.js, МП JavaScript і TypeScript та IDE Visual Studio Code для розробки серверу системи.

4 КОНСТРУЮВАННЯ ОНЛАЙН ВІРТУАЛЬНОГО СЕРЕДОВИЩА РОЗУМНИЙ БУДИНОК ДЛЯ ГІБРИДНОЇ ЛАБОРАТОРІЇ GOLDI

4.1 Розробка алгоритму функціонування системи

Алгоритм функціонування системи можна представити у вигляді діаграми активності, наведеної на рисунку 4.1.

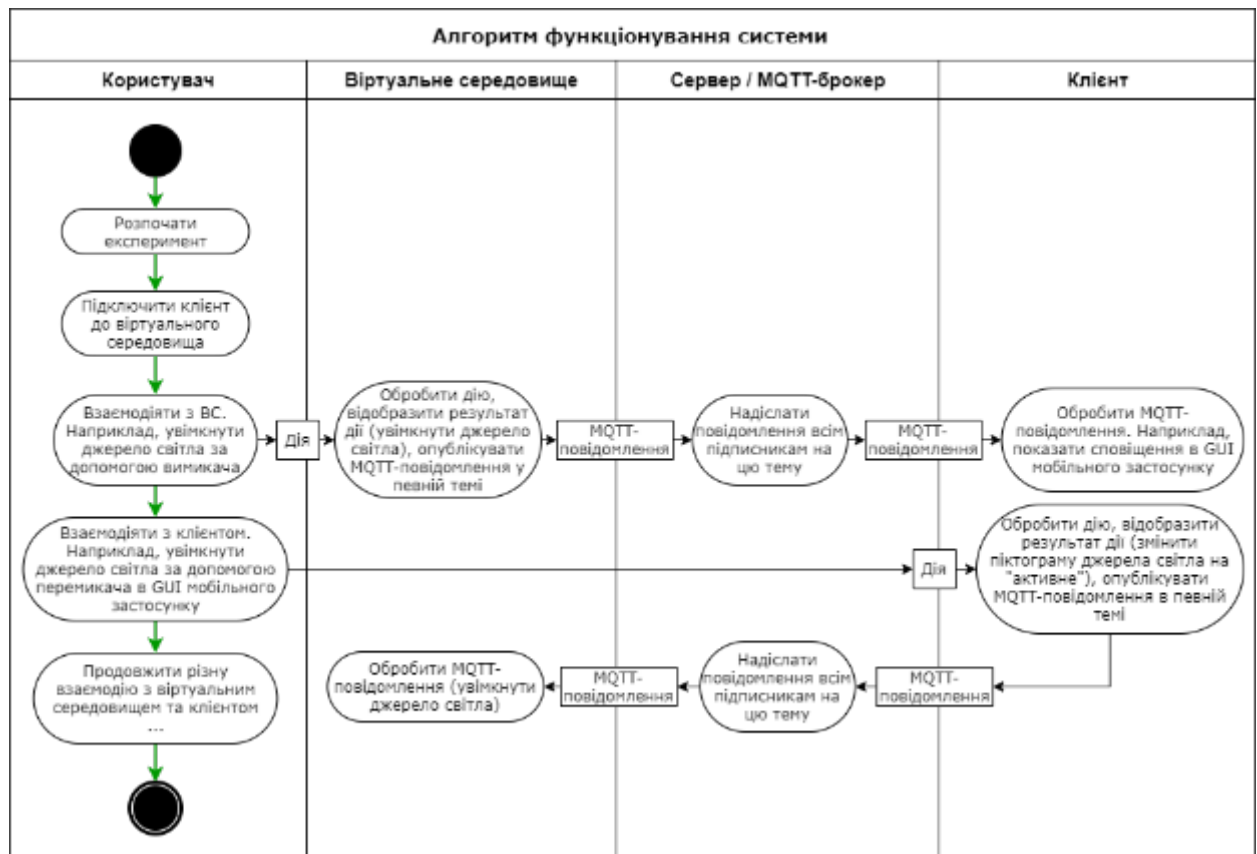


Рисунок 4.1 – Алгоритм функціонування системи

Насамперед користувач повинен розпочати експеримент та підключити свої ПАК до ВС (він не обов'язково повинен їх підключати, проте без цього втрачається сенс проведення експерименту). Після цього він може розпочати взаємодію з ВС та його обладнанням. Наприклад, коли користувач обирає вимикач джерела світла, тоді ВС наказує ігровому агенту рухатися до нього. Після цього користувач натискає F для взаємодії з вимикачем. Оскільки агент вже поруч з вимикачем, відбувається взаємодія з ним та вмикається прив'язане

до нього джерело світла. Після цього у відповідну тему даного вимикача публікується MQTT-повідомлення про те, що джерело світла було увімкнено. Отримане повідомлення MQTT-брокер, який запущено на сервері, розсилає всім підписникам на цю тему. Користувацькі ПАК-підписники отримують дане повідомлення та можуть відобразити зміну стану джерела світла у своєму інтерфейсі. Наприклад, ПАК у вигляді мобільного застосунку може показати сповіщення або змінити піктограму джерела світла на «активне».

Крім цього, користувач може взаємодіяти зі своїми під'єднаними до ВС ПАК. Наприклад, за допомогою перемикача у своєму мобільному застосунку він може увімкнути джерело світла. Піктограма цього джерела світла зміниться на «активне» у GUI даного ПАК. Після цього ПАК повинен у відповідну тему вимикача даного джерела світла опублікувати MQTT-повідомлення про те, що джерело світла було увімкнено. Отримане повідомлення MQTT-брокер розсилає всім підписникам на цю тему. ВС отримує це повідомлення та вмикає відповідне джерело світла.

Алгоритм підключення ПАК користувачів до ВС представлено на рисунку 4.2. Насамперед користувач повинен перейти на вебсторінку ВС, після чого ВС запропонує розпочати експеримент. Після натискання користувачем кнопки «Почати» ВС надсилає запит до API серверу для початку сеансу. Сервер обробляє отриманий запит, генерує ІС, запускає MQTT-брокер для сеансу та зв'язує з ідентифікатором сеансу ВС та брокер. У відповідь на запит сервер повертає хост, порт, ІС та деяку іншу інформацію щодо сеансу (див. п.п. 3.3.1). ВС відображає отримані хост, порт та ІС за допомогою HUD, після чого починає процес під'єднання свого обладнання до MQTT-брокеру. Після вдалого підключення обладнання до брокеру ВС запускає таймер сеансу (одна година).

Користувач повинен відображені хост, порт та ІС ввести в клієнті. Після цього ПАК повинен підключитися до MQTT-брокеру за введеними хостом та портом. Після підключення ПАК обов'язково повинен підписатися на вітальну тему, у назві якої міститься ІС. Сервер отримує повідомлення від

MQTT-брокеру сеансу про підписку клієнта, після чого сервер перевіряє, чи вказано правильний ІС в темі підписки. Якщо ідентифікатор правильний, сервер зв'язує клієнта з його ВС за ідентифікатором сеансу та відправляє їм обом MQTT-повідомлення про успішне підключення ПАК. ВС відображає користувачу отримане повідомлення про успішне підключення клієнта.

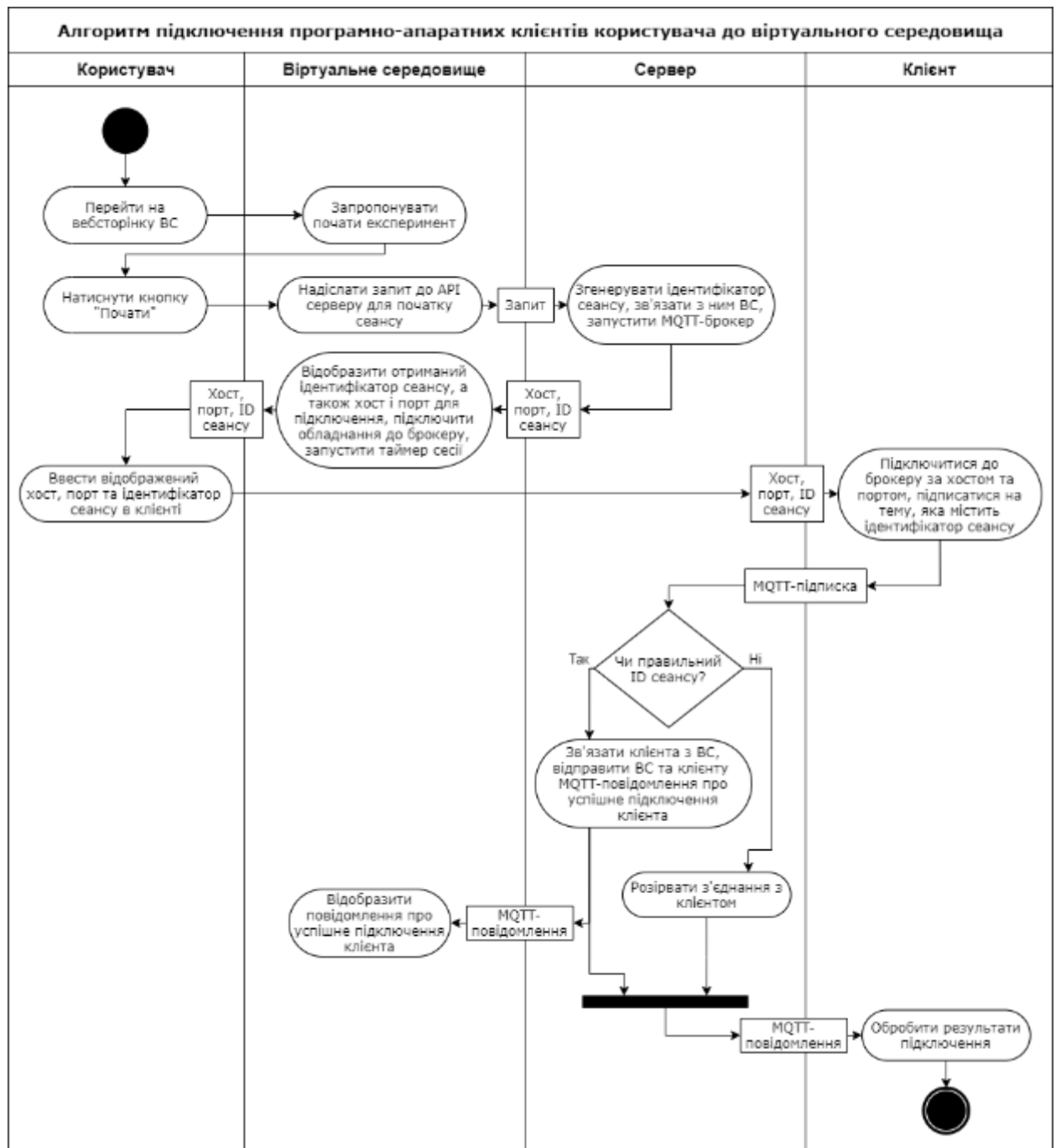


Рисунок 4.2 – Алгоритм підключення ПАК користувача до ВС

Якщо ідентифікатор неправильний, сервер наказує MQTT-брокеру розірвати з'єднання з ПАК. Після підписки на вітальну тему з ідентифікатором сеансу ПАК у будь-якому випадку отримає MQTT-повідомлення про результат, яке потім обробляє у той спосіб, у який його запрограмував користувач.

Отже, основними етапами роботи розроблюваної системи є: початок експерименту (надсилання віртуальним середовищем запиту до API серверу для початку сеансу, генерація сервером ІС та запуск MQTT-брокеру), підключення ПАК користувача до ВС (аутентифікація ПАК), взаємодія з ВС через GUI та дистанційно за допомогою ПАК користувача.

4.2 Опис особливостей програмної реалізації

4.2.1 Особливості програмної реалізації серверу

У розробленому сервері можна виділити наступні основні типи модулів:

- класи-сервіси – це класи, які реалізують основний функціонал серверу. До цього модулю належать такі класи, як: `SessionsService`, `BrokerService` та `DBService`;

- роутери – це модулі, які реалізують обробку запитів конкретного типу, викликаючи функції класів-сервісів та повертаючи отримані результати як відповідь сервера. Зараз наявний лише один роутер – `sessionRouter`, проте якщо в майбутньому, наприклад, знадобиться реалізувати підтримку акаунтів користувачів, тоді для обробки таких запитів можна буде створити роутер `userRouter`;

- утиліти – це модулі, які реалізують додатковий незначний функціонал. Наприклад, функції виведення логів у термінал та функції, які отримують дані, а повертають відформатований JSON-об'єкт з цими даними тощо;

– константи – це модуль, який містить усі важливі значення, константи та строки-повідомлення для забезпечення можливості швидкої зміни конфігурації серверу. Наприклад, значення довжини ІС, термін його дійсності (зараз це значення дорівнює одній годині) тощо.

Як було відзначено у підрозділі 3.2, на сервері повинні бути реалізовані механізми захисту. Для захисту HTTP-заголовків відповідей сервера на запити було використано модуль `Helmet` для `Node.js`. Він прибирає небезпечні заголовки, які можуть виказати інформацію, наприклад, про те, що використовується фреймворк `Express.js`. Така інформація могла б допомогти зловмиснику, оскільки, знаючи використовуваний фреймворк, він міг би використати його вразливості для завдання шкоди серверу. Крім цього, у запитів, які обов'язково мають вхідний параметр у вигляді ІС, роутери обов'язково перевіряють тип параметру та, якщо параметр має тип строки, його відповідність встановленій конфігурацією серверу довжині ІС [105].

Для обмеження кількості запитів з одного ІР за період часу було використано модуль `Express Rate Limit`. Було вирішено блокувати доступ до серверу з ІР, якщо протягом 15 хвилин з нього надійшло більше ста запитів [106].

Сервер було спроектовано так, що у разі його аварійного завершення БД залишається в останньому стабільному стані, а в разі запланованого завершення серверу БД очищається. При кожному запуску утиліта `startServer` перевіряє, чи є дані про сеанси в БД і, якщо такі дані є, відновлює ці сеанси і екземпляри ВС можуть поновити з'єднання зі своїми MQTT-брокерами.

У процесі запуску нового сеансу відбувається багато операцій. Насамперед перевіряється, чи не перевищено ліміт кількості одночасно активних сеансів. Для генерації ІС використовується модуль `Uuid`, який генерує універсально унікальний ідентифікатор потрібної довжини. Згенерований ідентифікатор додатково перевіряється на відсутність співпадіння з ідентифікаторами активних сеансів. Після цього відбувається

пошук вільного порту для запуску MQTT-брокера (було обрано модуль Aedes) [107]-[108].

Основні методи та змінні серверу наведено в таблиці 4.1.

Таблиця 4.1 – Основні методи та змінні серверу

Назва	Тип	Належність	Призначення
1	2	3	4
const REQUESTS_LIMIT_PERIOD	Константа	server Config	Встановлює часовий інтервал, протягом якого кількість запитів з одного IP не повинна перевищувати значення REQUESTS_LIMIT_PERIOD
const REQUESTS_LIMIT_PER_PERIOD	Константа	server Config	Встановлює обмеження кількості запитів з одного IP за інтервал часу REQUESTS_LIMIT_PERIOD
const SESSION_ID_LENGTH	Константа	server Config	Визначає кількість символів в IC
const SESSION_TIMELIFE	Константа	server Config	Зберігає значення часу «життя» сеансу
const SESSIONS_LIMIT	Константа	server Config	Визначає обмеження кількості одночасно активних сеансів
private sessions: Map<string, BrokerService>	Поле класу	Sessions Service	Зберігає об'єкти BrokerService активних сеансів
private timers: Map<string, NodeJS.Timeout>	Поле класу	Sessions Service	Зберігає таймери активних сеансів
async loadAllSessions()	Метод класу	Sessions Service	Запускає всі незавершені сеанси, дані про які наявні в БД
async startSession(session_id?: string, host: string = HOST, clientsPort?: number)	Метод класу	Sessions Service	Запускає новий сеанс або сеанс для незавершеного сеансу з БД
async reconnect(session_id: string)	Метод класу	Sessions Service	Перезапускає вказаний сеанс
async closeAllSessions(needTo RemoveFromDB = true)	Метод класу	Sessions Service	Завершує всі сеанси, опціонально видаляючи дані про них з БД
startSessionTimer(session_id: string)	Метод класу	Sessions Service	Запускає таймер вказаного сеансу
continueSession(session_id: string)	Метод класу	Sessions Service	Подовжує час вказаного сеансу
closeSession(session_id: string, needToRemove FromDB = true)	Метод класу	Sessions Service	Завершує вказаний сеанс, опціонально видаляючи дані про нього з БД

Продовження таблиці 4.1

1	2	3	4
private async findFreeUniquePort(port?: number undefined)	Метод класу	Sessions Service	Шукає вільний порт для запуску MQTT-брокеру
private broker!: Aedes	Поле класу	Broker Service	Зберігає MQTT-брокер сеансу
private clients: Map<string, AedesClient>	Поле класу	Broker Service	Зберігає підключенні до брокеру клієнти
private session_id: string	Поле класу	Broker Service	Зберігає ІС
private host: string	Поле класу	Broker Service	Зберігає хост сеансу
private clientsPort: number	Поле класу	Broker Service	Зберігає порт сеансу
address(): IBrokerAddressInfo	Метод класу	Broker Service	Повертає адресу брокера
close(callback?: (err?: Error) => void): void	Метод класу	Broker Service	Завершує брокер
private initBroker(): void	Метод класу	Broker Service	Запускає брокер
private db	Метод класу	DBService	Містить об'єкт БД
saveSession(session_id: string, host: string, clientsPort: number)	Метод класу	DBService	Зберігає дані про сеанс у БД
findSessionById(session_id: string)	Метод класу	DBService	Шукає у БД сеанс за ідентифікатором
updateSession(session_id: string, newValues: IUpdateSession)	Метод класу	DBService	Оновлює дані сеансу у БД
removeSession(session_id: string)	Метод класу	DBService	Видаляє сеанс з БД
loadAllSessions()	Метод класу	DBService	Повертає записи про всі сеанси з БД
cleanSessions()	Метод класу	DBService	Видаляє записи про всі сеанси з БД
const sessionActionsHandler = async (req: Request, res: Response, action: () => Promise<any>, jsonObjectName: string): Promise<void>	Функція	Утиліти	Універсальний обробник запитів, який перевіряє ІС, виконує передану дію та повертає результат як відповідь на запит
const startServer = (server: Express, port: number): void	Функція	Утиліти	Перевіряє, чи є дані про сеанси в БД і, якщо такі дані є, відновлює ці сеанси
const isSessionIdValid = (session_id: string): boolean	Функція	Утиліти	Перевіряє, чи отриманий у запиті ІС має правильний формат

Отже, можна виділити такі основні функції розробленого серверу: методи SessionsService::startSession, SessionsService::startSessionTimer, SessionsService::reconnect, SessionsService::continueSession, BrokerService::initBroker, DBService::saveSession, DBService::removeSession; утиліти sessionActionsHandler, startServer тощо.

4.2.2 Особливості програмної реалізації віртуального середовища

У розробленому ВС можна виділити наступні основні типи класів:

- класи акторів – це класи, які мають візуальне подання та представляють об’єкти, які можуть бути розміщені на 3D-сцені мапи, а також реалізують логіку їх поведінки та взаємодії з іншими об’єктами. До таких класів належать: абстрактні базові класи BP_Equipment, BP_RemoteControlledEquipment, BP_Sensor; класи BP_Room, BP_Door, BP_Window, BP_Switch, BP_ClimateController, BP_LightSource, BP_ClimateControlSystem, BP_LightnessSensor, BP_FencedGround тощо;
- віджети – це класи компонентів GUI, які можуть відображатися на екрані поверх 3D-сцени в процесі гри. До таких класів належать: W_HUD, W_Tooltip, W_SessionFinishNotifier, W_Loader, W_ErrorMessage, W_AskWhatToDoWithSession, W_ClimateController, W_ClientsConnectionData тощо;
- віджети-утиліти редактора – це класи компонентів GUI, які не використовуються під час гри, а потрібні для реалізації віджетів для IDE UE4. Тобто такі віджети використовуються в процесі розробки гри для полегшення певних процесів розробки на 3D-сцені гри. Наприклад, за допомогою таких віджетів можна реалізувати функціонал швидкого розміщення акторів на сцені, дотримуючись певних правил на кшталт висоти розташування над рівнем землі. Даний тип класу у розробленому ПП представлений класом EUW_RoomsConnector, який допомагає з’єднувати окремі кімнати на мапі в єдиний будинок;

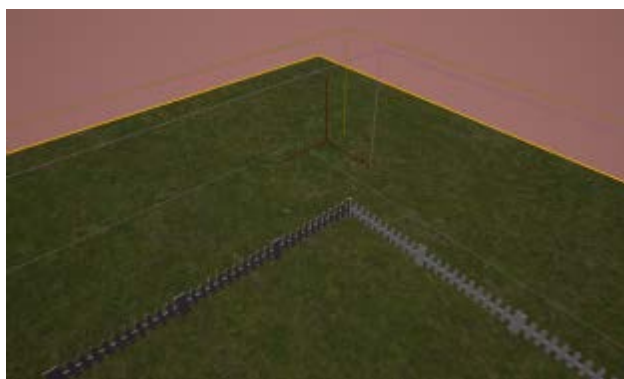
– класи-контролери – це класи, які не мають графічного подання, проте використовуються для контролю ігрового процесу або ігрового персонажу. До таких класів належать BP_GameInstance та BP_TopDownController;

– класи-утиліти – це класи, які реалізують допоміжний функціонал. Наприклад, клас BP_ConfigReader, який реалізує зчитування з файлу конфігурації для підключення до REST API серверу, та клас BP_Transmitter, який реалізує функціонал комунікації з REST API серверу;

– класи-структури – це класи, які використовуються для передачі структур даних між різними класами. Наприклад, клас BP_MqttMessage, який зберігає тему та текст MQTT-повідомлення, клас BP_LogListItemData, який зберігає повідомлення та тип логу для відображення у журналі подій.

Було реалізовано функції параметричної побудови поверхні підвір'я та паркану, а також кімнат для забезпечення програмістам ВС можливості подальшої перебудови 3D-сцени ВС для створення нових конфігурацій «розумного» будинку.

Клас BP_FencedGround представляє поверхню землі з парканом по периметру та додатковою недосяжною областю за ним для того, щоб з позиції камери спостереження за персонажем не було видно порожнечі поза картою (рис. 4.3).



а



б

а) вигляд у редакторі 3D-сцени; б) вигляд під час гри

Рисунок 4.3 – Вигляд об'єкту класу BP_FencedGround

Після того як розміри поверхні було змінено у редакторі 3D-сцени, BP_FencedGround насамперед викликає метод SetGroundScaleToFit для розрахунку розміру поверхні. Для цього розраховується необхідна кількість секцій паркану для кожної сторони поверхні за формулою (4.1):

$$N_x = \frac{W_{gx}}{L}, \quad (4.1)$$

де N_x – необхідна кількість секцій паркану для сторони поверхні;

W_{gx} – довжина сторони поверхні, м;

L – довжина секції паркану, яка для використовуваної 3D-моделі секції паркану дорівнює 5,22 м.

Отримані значення N_a та N_b є дійсними числами, тому їх потрібно округлити та розрахувати нові параметри масштабування поверхні, оскільки в UE4 розміри можливо змінювати тільки у відносних одиницях, тобто неможливо вказати абсолютне значення довжини. Для цього розраховуються нові значення масштабування для кожної сторони поверхні за формулою (4.2):

$$S_{gx} = \frac{N_x \cdot L}{40}, \quad (4.2)$$

де S_{gx} – значення масштабування для сторони поверхні.

Для розміщення N_x секцій паркану вздовж сторони поверхні викликається метод AddNewFenceSection, у якому розраховуються координати для розміщення кожної секції. Усі координати точки розміщення відносно поверхні дорівнюють нулю, окрім координати, вздовж координатної осі якої знаходиться сторона. Для розрахунку значення координати секції на цій осі використовується формула (4.3):

$$C_i = \frac{L \cdot i}{S_{gx}}, \quad (4.3)$$

де C_i – координата вздовж осі сторони поверхні;

$$i = 0, 1, \dots, N_x - 1.$$

Після розміщення секція обертається у потрібному напрямку.

Розміри кімнат програмісти можуть вказати за допомогою налаштувань об'єкту класу BP_Room (рис. 4.4), які він може змінити у редакторі 3D-сцени.

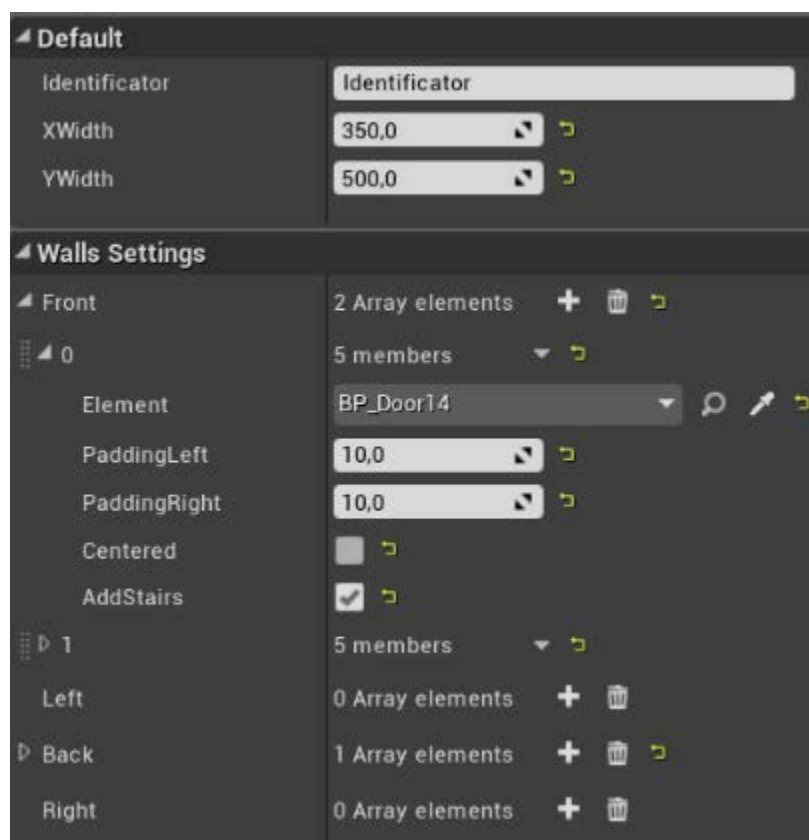


Рисунок 4.4 – Приклад налаштувань об'єкту класу BP_Room у редакторі 3D-сцени

Розрахунок значень масштабування кожної сторони основи кімнати відбувається за формулою (4.4):

$$S_{rx} = \frac{L_{px}}{4}, \quad (4.4)$$

де S_{rx} – значення масштабування для сторони кімнати;

L_{px} – довжина сторони кімнати, яку вказав програміст у налаштуваннях, м.

Після масштабування основи перевіряється можливість побудови стін за вказаною у налаштуваннях кімнати їх конфігурацією. Для цього вказане значення довжини відповідної сторони кімнати порівнюється з сумою розмірів доданих до цієї стіни елементів за формулою (4.5):

$$W_{rx} = \sum_{i=0}^{N_e-1} E_i + PL_i + PR_i, \quad (4.5)$$

де W_{rx} – довжина усіх елементів стіни, м;

N_e – кількість елементів стіни;

E_i – довжина потрібної секції стіни, м;

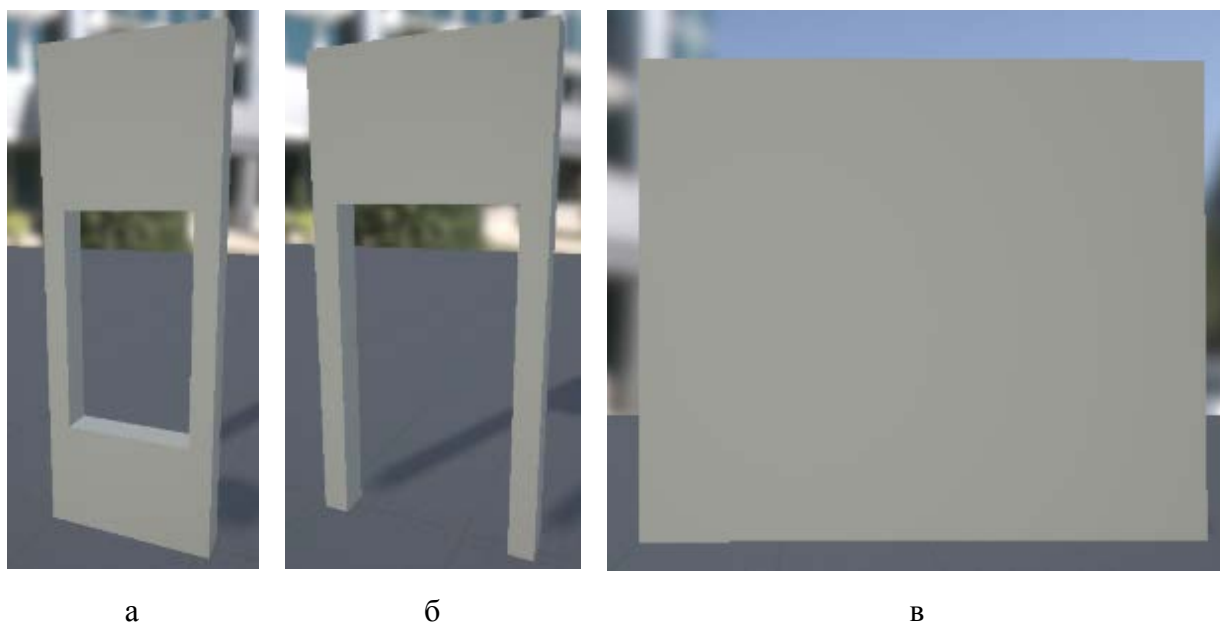
PL_i – відступ ліворуч, м;

PR_i – відступ праворуч, м.

Якщо отримане для стіни значення W_{rx} менше, ніж L_{px} для цієї сторони кімнати, тоді ця стіна може бути згенерована.

E_i може мати одне з трьох можливих значень у залежності від типу секції (рис. 4.5): E_i секції для вікна (рис. 4.5а) дорівнює 1,08 м; E_i секції для дверей (рис. 4.5б) дорівнює 1,46 м; E_i цільної секції (рис. 4.5в) дорівнює 4 м, проте при генерації ця секція масштабується під потрібну довжину за формулою (4.4), де L_{px} – це значення потрібної довжини секції.

Після того як об'єкт BP_Room було додано на мапу та він згенерував свою 3D-модель подання кімнати, об'єкт перевіряє акторів, які перетинаються з ним. Якщо це актор класу-спадкоємця BP_Equipment і його поле ParentRoom вільне, тоді BP_Room зберігає його у своєму полі-переліку обладнання StuffList, а у поле ParentRoom даного обладнання зберігається посилання на дану кімнату.



а) секція для вікна; б) секція для дверей; в) цільна секція

Рисунок 4.5 – Типи секцій, з яких складається стіна

Моделювання показників, які передає датчик температури, залежить від трьох залежних від часу змінних, які задають теплообмін у кімнаті: температура кімнати (T_{room}); надходження теплоти, тобто передача теплової енергії від нагрівача (Q_{gain}) до кімнати; втрата тепла, тобто передача теплової енергії від кімнати (Q_{loss}) до зовнішнього середовища [109].

Диференціальне рівняння визначає зв'язок між цими змінними, але оскільки теплопередача задана в термінах зміни температури, тільки кімнатна температура є змінною стану [109].

Рівняння надходження теплоти від нагрівача представлене формулою (4.6):

$$\frac{dQ_{gain}}{dt} = M_{heaterair} \cdot c_{air} \cdot (T_{heater} - T_{room}), \quad (4.6)$$

де $\frac{dQ_{gain}}{dt}$ – швидкість надходження теплоти;

$M_{heaterair}$ – постійний рівень маси повітря, що проходить через нагрівач, дорівнює 1 кг/с;

c_{air} – питома теплоємність, 1005,4 Дж/кг·°С;

T_{heater} – постійна температура повітря від нагрівача, дорівнює 50°С;

T_{room} – початкова температура кімнати, дорівнює 20°С [109].

Теплова енергетична втрата кімнати через теплопровідність стін та вікна, яка пропорційна перепаду температур між кімнатною температурою та зовнішньою температурою, визначається формулою (4.7):

$$\frac{dQ_{loss}}{dt} = \frac{(T_{room} - T_{outside})}{R}, \quad (4.7)$$

де $\frac{dQ_{loss}}{dt}$ – швидкість втрати теплоти;

$T_{outside}$ – зовнішня температура, дорівнює 25°С;

R – тепловий спротив, дорівнює $4,329 \cdot 10^{-7}$ К/Вт [109].

Рівняння зміни температури у кімнаті під дією нагрівача визначається за формулою (4.8):

$$\frac{dT_{room}}{dt} = \frac{1}{m_{roomair} \cdot c_{air}} \cdot \left(\frac{dQ_{gain}}{dt} - \frac{dQ_{loss}}{dt} \right), \quad (4.8)$$

де $\frac{dT_{room}}{dt}$ – швидкість зміни температури у кімнаті;

$m_{roomair}$ – постійний рівень маси повітря у кімнаті, дорівнює 1470 кг [109].

Основні методи та змінні віртуального середовища наведено в таблиці 4.2.

Таблиця 4.2 – Основні методи та змінні віртуального середовища

Назва	Тип	Належність	Призначення
1	2	3	4
Identifier	Поле	BP_Equipment	Зберігає ідентифікатор обладнання
State	Поле	BP_Equipment	Зберігає стан обладнання
Description	Поле	BP_Equipment	Зберігає опис обладнання

Продовження таблиці 4.2

1	2	3	4
TooltipWidget	Поле	BP_Equipment	Зберігає посилання на віджет для відображення ідентифікатора та опису
ParentRoom	Поле	BP_Equipment	Зберігає посилання на кімнату, до якої належить
Interact	Метод	BP_Equipment	Визначає поведінку актора у разі взаємодії ігрового персонажу з ним
GetIdentificator	Метод	BP_Equipment	Повертає ідентифікатор та тип обладнання
SetHighlighterForSM	Метод	BP_Equipment	Реалізує підсвічування компонентів обладнання, коли актор активний
SetTooltipVisibility	Метод	BP_Equipment	Відображає віджет з ідентифікатором та описом обладнання
SetTooltipPointVisibility	Метод	BP_Equipment	Відображає мітку-індикатор, яка вказує на те, що обладнання має ідентифікатор чи опис
SetTranslucency	Метод	BP_Equipment	Встановлює режим відображення актора – прозорий / непрозорий
MqttClient	Поле	BP_Remote ControlledEquipment	Зберігає посилання на власний MQTT-клієнт
Topic	Поле	BP_Remote ControlledEquipment	Зберігає MQTT-тему, у яку має публікувати повідомлення
ConnectionStatus	Поле	BP_Remote ControlledEquipment	Зберігає стан з'єднання
PublishMessage	Метод	BP_Remote ControlledEquipment	Публікує MQTT-повідомлення
ConnectToBroker	Метод	BP_Remote ControlledEquipment	Встановлює з'єднання з MQTT-брокером
DisconnectFromBroker	Метод	BP_Remote ControlledEquipment	Від'єднується від MQTT-брокеру
OnReceivedData	Метод	BP_Remote ControlledEquipment	Обробляє вхідне MQTT-повідомлення
DoHandshake	Метод	BP_Remote ControlledEquipment	Підписується на вітальну тему (див. п.п. 3.3.2)
HandleMessage	Метод	BP_Remote ControlledEquipment	Обробляє команду, отриману у MQTT-повідомленні
OnError	Метод	BP_Remote ControlledEquipment	Обробляє помилки MQTT-клієнту
OnClosed	Метод	BP_Remote ControlledEquipment	Обробляє завершення MQTT-клієнту
SetConnectionHighlighter OfActor	Метод	BP_Remote ControlledEquipment	Реалізує підсвічування компонентів обладнання в залежності від статусу з'єднання

Продовження таблиці 4.2

1	2	3	4
UpdatePeriod	Поле	BP_Sensor	Зберігає інтервал, з яким датчик повинен відправляти MQTT-повідомлення з вимірюваннями
Update	Метод	BP_Sensor	Відправляє вимірювання
Identificator	Поле	BP_Room	Зберігає ідентифікатор кімнати
XWidth	Поле	BP_Room	Зберігає довжину сторони X
YWidth	Поле	BP_Room	Зберігає довжину сторони Y
HUD	Поле	BP_Room	Зберігає посилання на HUD для передачі деяких даних
FrontWall	Поле	BP_Room	Зберігає перелік елементів передньої стіни
LeftWall	Поле	BP_Room	Зберігає перелік елементів лівої стіни
BackWall	Поле	BP_Room	Зберігає перелік елементів задньої стіни
RightWall	Поле	BP_Room	Зберігає перелік елементів правої стіни
StuffList	Поле	BP_Room	Зберігає перелік посилань на обладнання
EquipmentNumber	Поле	BP_Room	Зберігає кількість обладнання у кімнаті
CreateWall	Метод	BP_Room	Генерує стіну для вказаної сторони кімнати
CalculateWallBlockLocation	Метод	BP_Room	Розраховує позицію секції стіни
CalculateSolidWallScale	Метод	BP_Room	Розраховує масштабування цільної секції стіни
CheckIsEnoughSpace	Метод	BP_Room	Перевіряє, чи вистачить місця для генерації потрібної стіни
SpawnRequiredWallBlock	Метод	BP_Room	Додає потрібну секцію стіни у модель кімнати
ConnectEquipmentToBroker	Метод	BP_Room	Наказує обладнанню підключитися до брокера, якщо воно це реалізує
DisconnectEquipmentFromBroker	Метод	BP_Room	Наказує обладнанню від'єднатися від брокера, якщо воно це реалізує
GrassScale	Поле	BP_FencedGround	Зберігає масштабування текстури трави
UnreachableAreaScale	Поле	BP_FencedGround	Зберігає розмір недосяжної для персонажу додаткової зони
FenceSectionLength	Поле	BP_FencedGround	Зберігає довжину секції паркану
FenceSectionNumberByX	Поле	BP_FencedGround	Зберігає кількість секцій паркану для сторони X
FenceSectionNumberByY	Поле	BP_FencedGround	Зберігає кількість секцій паркану для сторони Y

Продовження таблиці 4.2

1	2	3	4
AddNewFenceSection	Метод	BP_FencedGround	Додає нову секцію паркану
SetGroundScaleToFit	Метод	BP_FencedGround	Встановлює новий масштаб поверхні
CalculateFenceSectionNumberForSides	Метод	BP_FencedGround	Розраховує необхідну кількість секцій паркану для сторони
SetGroundMaterial	Метод	BP_FencedGround	Встановлює масштабування текстури трави
Events	Поле	W_HUD	Елемент для відображення журналу подій
PublishedMessages	Поле	W_HUD	Елемент для відображення журналу опублікованих MQTT-повідомлень
ConnectionData	Поле	W_HUD	Елемент для відображення даних для під'єднання ПАК до брокеру
AddEventsLog	Метод	W_HUD	Додає новий запис у журнал подій
AddPublishedMessagesLog	Метод	W_HUD	Додає новий запис у журнал опублікованих повідомлень
UpdateHUDWithConnectionData	Метод	W_HUD	Оновлює дані для під'єднання ПАК до брокеру
Rooms	Поле	BP_GameInstance	Зберігає перелік кімнат
Timer	Поле	BP_GameInstance	Зберігає таймер
RoomsNumber	Поле	BP_GameInstance	Зберігає кількість кімнат
isConnected	Поле	BP_GameInstance	Зберігає статус з'єднання
ConnectedRoomsNumber	Поле	BP_GameInstance	Зберігає кількість кімнат, у яких більша частина обладнання під'єдналася до брокеру
HUD	Поле	BP_GameInstance	Зберігає посилання на HUD
Loader	Поле	BP_GameInstance	Зберігає посилання на віджет для індикації завантаження
Start	Метод	BP_GameInstance	Розпочинає експеримент
StartTimer	Метод	BP_GameInstance	Запускає таймер
ReadConnectionConfiguration	Метод	BP_GameInstance	Отримує конфігурацію серверу для з'єднання за допомогою BP_ConfigReader
ConnectToServer	Метод	BP_GameInstance	Підключається до API серверу за допомогою BP_Transmitter
FinishSession	Метод	BP_GameInstance	Завершує сеанс
FileName	Поле	BP_ConfigReader	Зберігає ім'я файлу з конфігурацією
ReadConfig	Метод	BP_ConfigReader	Зчитує конфігурацію
SaveConfig	Метод	BP_ConfigReader	Зберігає конфігурацію у файл
ApiHost	Поле	BP_Transmitter	Зберігає хост API серверу
ApiPort	Поле	BP_Transmitter	Зберігає порт API серверу
MqttHost	Поле	BP_Transmitter	Зберігає хост брокеру

Продовження таблиці 4.2

1	2	3	4
MqttPort	Поле	BP_Transmitter	Зберігає порт брокеру
SessionID	Поле	BP_Transmitter	Зберігає ІС
StartSession	Метод	BP_Transmitter	Відправляє запит для початку сеансу
StartSessionTimer	Метод	BP_Transmitter	Відправляє запит для запуску таймеру сеансу
ContinueSession	Метод	BP_Transmitter	Відправляє запит на продовження часу сеансу
ReconnectToSession	Метод	BP_Transmitter	Відправляє запит на повторне підключення до сеансу
FinishSession	Метод	BP_Transmitter	Відправляє запит на завершення сеансу
MinInteractionDistance	Поле	BP_TopDown Controller	Містить значення мінімальної відстані до обладнання для початку взаємодії
MoveToHitLocation	Метод	BP_TopDown Controller	Починає рух до вказаного місця
CalculateDistanceFrom PlayerToLocation	Метод	BP_TopDown Controller	Розраховує відстань від персонажу до вказаного місця

Отже, можна виділити основні групи методів, що були реалізовані у ВС: методи для зчитування конфігурації, методи для комунікації з API серверу, методи для комунікації з MQTT-брокером, методи для керування сеансом, методи Interact класів обладнання, методи моделювання показників датчиків, методи генерації моделей підвір'я та кімнати, метод MoveToHitLocation для переміщення ігрового персонажу тощо.

4.3 Опис структури даних

Як вже було зазначено у підрозділі 3.2, БД SQLite було обрано для зберігання на сервері даних про активні сеанси. Для зберігання цих даних достатньо однієї таблиці sessions, структуру якої наведено на рисунку 4.6.

Таблиця sessions має наступні поля:

- id – це ключове поле, яке зберігає унікальний ідентифікатор запису у таблиці;

- session_id – це текстове поле, яке зберігає ІС. Зараз у конфігурації серверу (див. п.п. 4.2.1) значення довжини ІС дорівнює 7, тому довжина поля session_id у таблиці також дорівнює 7. Якщо у майбутньому це значення буде змінено, тоді потрібно буде створити нову міграцію БД. Міграція визначає правила переходу від однієї версії структури БД до іншої та забезпечує потрібні перетворення наявних даних;
- host – це текстове поле, яке зберігає хост MQTT-брокеру, з яким взаємодіє ВС сеансу та його ПАК. Хоча зараз значення host однакове для всіх сеансів, проте у майбутньому може з'явитися потреба у запуску MQTT-брокерів на окремих хостах;
- clientsPort – це цілочисельне поле, яке зберігає порт серверу, на якому запущено MQTT-брокер сеансу.

sessions		
id	integer	"id" integer NOT NULL
session_id	varchar(7)	"session_id" varchar(7) NOT NULL
host	varchar(23)	"host" varchar(23) NOT NULL
clientsPort	integer	"clientsPort" integer NOT NULL

Рисунок 4.6 – Структура таблиці sessions

У процесі роботи серверу поле clientsPort може бути модифіковано. Це може статися з двох причин – перезапуск серверу після аварійного завершення та повторне підключення ВС після втрати Інтернет-з'єднання. Оновлення значення даного поля потрібне для того, щоб запобігти запуску декількох процесів на одному порту серверу. Тому після перезапуску сервер запускає всі сеанси, дані про які зберігаються у БД, та шукає для них вільний порт. При повторному підключенні оновлення цього поля також потрібне, оскільки MQTT-брокер завершує свою роботу після втрати з'єднання з усім обладнанням ВС, хоча сеанс залишається активним. Тому після надходження від ВС до серверу запиту про повторне підключення запускається новий MQTT-брокер, для якого потрібно знайти вільний порт.

На рисунку 4.7 представлено структуру BPS_WallElement, яка використовується у ВС для налаштування секцій стін кімнат. Вона має наступні поля:

- Element – посилання на актора класу-наслідника класу BP_Equipment. При генерації стін перевіряється, чи це об'єкт класу BP_Door або BP_Window. Якщо ні, тоді буде додано звичайну цільну секцію стіни (рис. 4.5в);
- PaddingLeft та PaddingRight – це поля, які визначають значення відступу у метрах відповідно ліворуч та праворуч від актора Element;
- Centered – це булеве поле, яке вказує, що Element повинен знаходитися в центрі стіни. Значення цього поля береться до уваги, якщо стіна, до якої належить дана секція, немає інших секцій;
- AddStairs – це булеве поле, яке вказує, що потрібно додати сходи перед Element. Значення цього поля береться до уваги, якщо Element є об'єктом класу BP_Door.

Element	 BP Equipment
PaddingLeft	 Float
PaddingRight	 Float
Centered	 Boolean
AddStairs	 Boolean

Рисунок 4.7 – Структура BPS_WallElement

Отже, розроблені структури даних є простими у реалізації, завдяки чому їх підтримка та оновлення не потребує багато зусиль. Розроблена структура БД не містить зайвих даних, що позитивно відзначається на продуктивності роботи серверу, оскільки операції з додавання, отримання, оновлення та пошуку даних не займають багато часу.

4.4 Розробка графічного інтерфейсу користувача

Інтерфейс стартового екрану ВС зображено на рисунку 4.8.

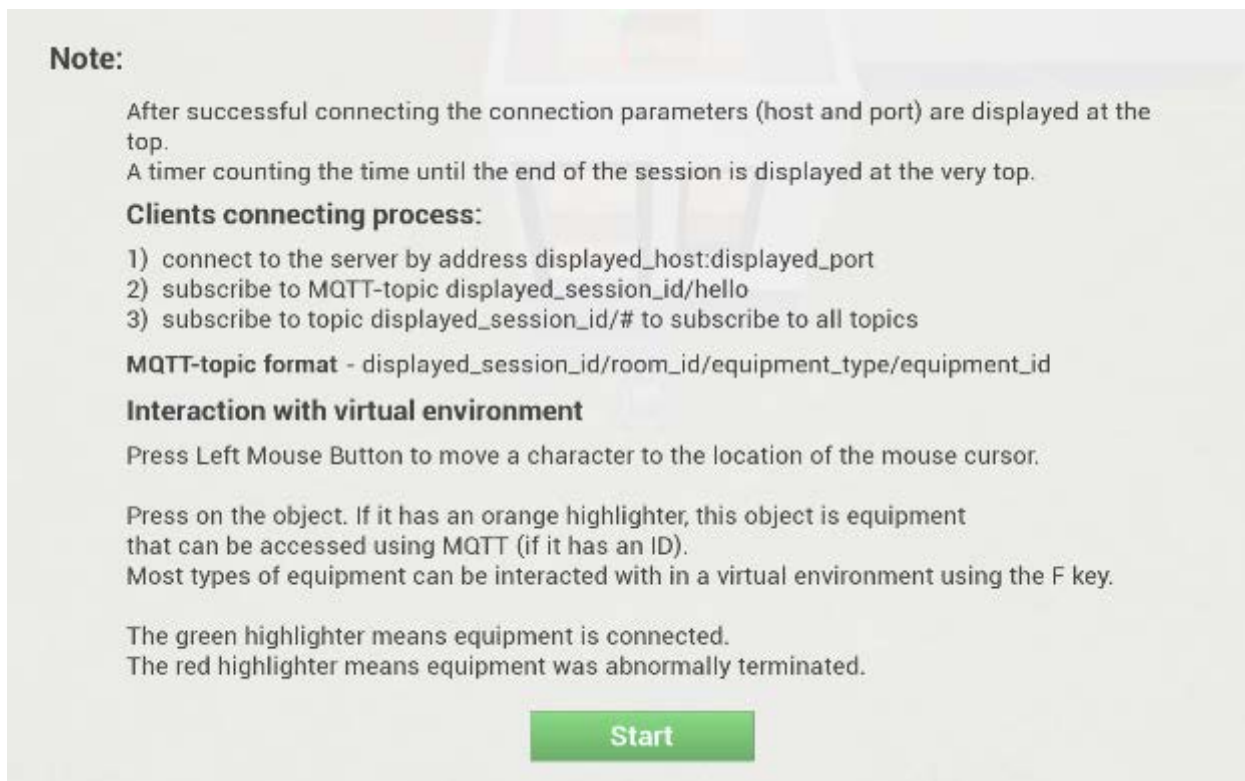


Рисунок 4.8 – Стартовий екран віртуального середовища

На даному екрані відображаються інструкції для користувача, які дають йому розуміння про те, які є основні елементи інтерфейсу та де вони відображаються, як підключити користувацькі ПАК та як взаємодіяти з ВС. Даний екран має ледь прозоре тло, а тому користувач може бачити будинок та ігрового персонажа.

Після натискання кнопки «Start» та успішного початку сеансу відображається будинок та його подвір'я. Вигляд ВС показано на рисунках 4.9 та 4.10.



Рисунок 4.9 – Вигляд будинку та його подвір'я з камери спостереження за персонажем



Рисунок 4.10 – Загальний вигляд віртуального середовища

Будинок – це поєднання декількох кімнат, які є об’єктами класу BP_Room (див. п. 3.2 та п.п. 4.2.2). Вигляд кімнат змінюється після того, як у зону дії їх триггеру потрапляє ігровий персонаж. На рисунку 4.11 зображено прозору кімнату. Коли персонаж потрапляє у зону дії триггеру, стають прозорими лише стіни кімнати, а обладнання залишається непрозорим. Це зроблено спеціально для того, щоб користувач чітко бачив обладнання у кімнаті, з яким він може взаємодіяти. Додатково над кожним обладнанням, з яким можна взаємодіяти, відображається індикатор у вигляді великого знаку оклику синього кольору (рис. 4.12-4.13).

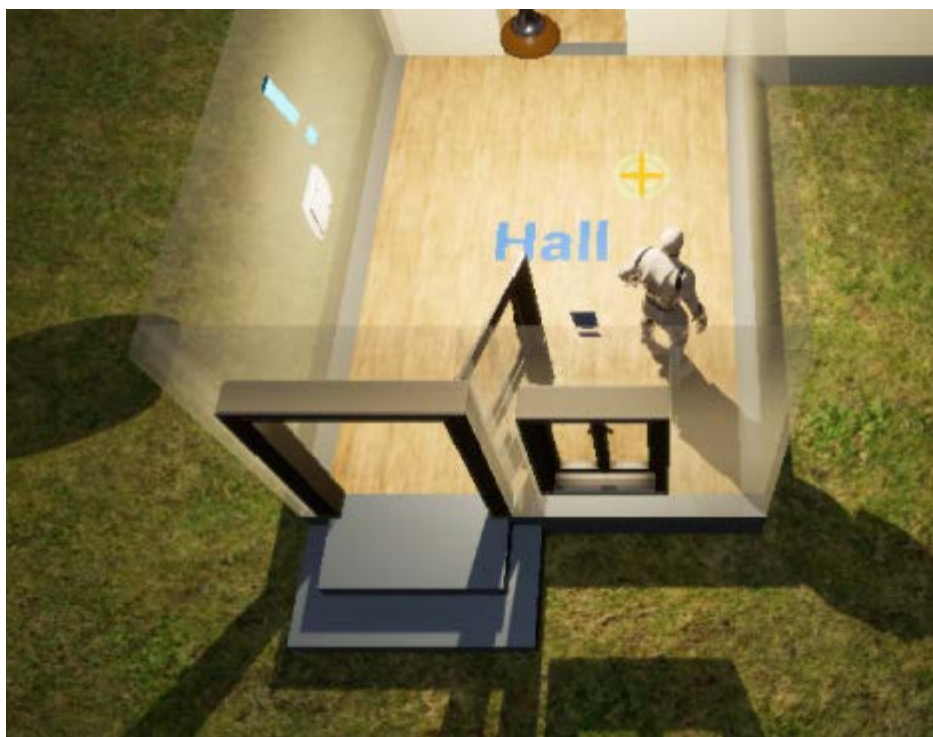


Рисунок 4.11 – Вигляд прозорої кімнати



Рисунок 4.12 – Індикатор обладнання, з яким можна взаємодіяти



Рисунок 4.13 – Кімната з різним обладнанням, з яким можна взаємодіяти

Обладнання також може ставати прозорим. Наприклад, коли персонаж знаходиться не у кімнаті, а стоїть на подвір'ї за нею, тоді обладнання стає прозорим разом з кімнатою, оскільки воно не має заважати спостереженню за персонажем (рис. 4.14).



Рисунок 4.14 – Вигляд прозорі кімнати та його обладнання

У кожній кімнаті на підлозі відображається її ідентифікатор (рис. 4.15). Це допомагає користувачу звертатися до обладнання, яке знаходиться у цій кімнаті, оскільки ідентифікатор кімнати є обов'язковою частиною MQTT-тем обладнання (див. п.п. 3.3.2).



Рисунок 4.15 – Ідентифікатор кімнати

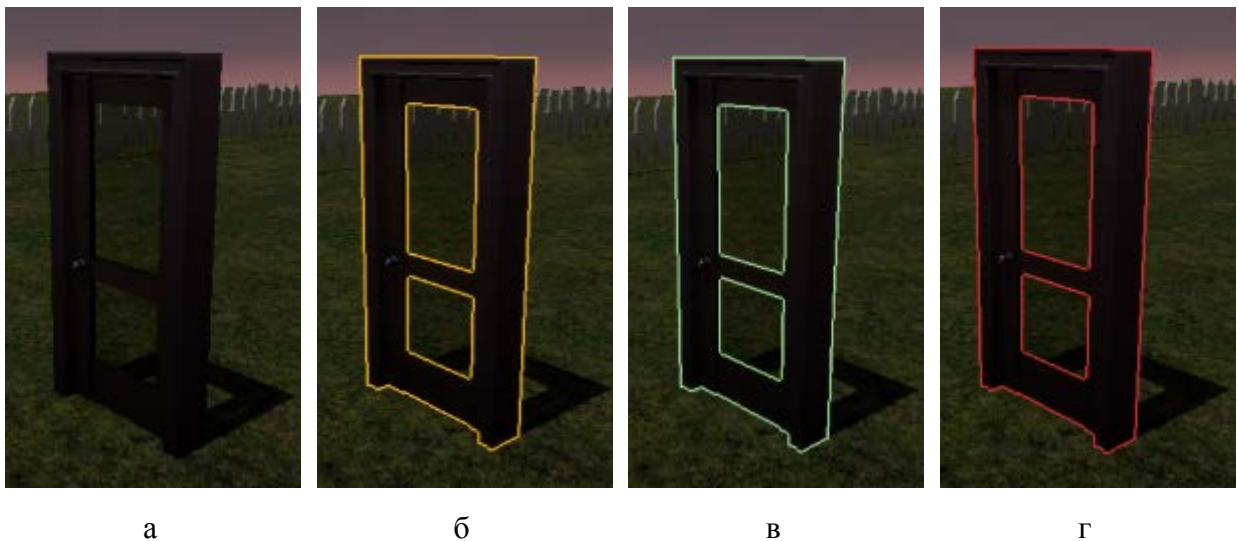
На рисунку 4.16 представлено HUD ВС. На ньому відображаються найнеобхідніші для користувача дані: таймер сеансу, параметри для підключення ПАК користувача (хост, порт, ІС), журнали подій та опублікованих обладнанням MQTT-повідомлень.



а) журнал подій; б) таймер сеансу; в) дані для підключення програмно-апаратних клієнтів користувача; г) журнал опублікованих обладнанням MQTT-повідомлень

Рисунок 4.16 – HUD віртуального середовища

На рисунку 4.17 представлено можливі типи виділення обладнання для індикації їх стану. Актор не має виділення (рис. 4.17а), коли він не є дистанційно керованим обладнанням або був спеціально відключений від брокера (наприклад, після завершення сеансу) та не вибраний користувачем. Обладнання має помаранчеве виділення (рис. 4.17б), коли воно активне, тобто користувач обрав його. Зелене та червоне виділення (відповідно рис. 4.17в та 4.17г) є індикатором стану з'єднання з MQTT-брокером – підключено / відключено через помилку або втрату зв'язку.



а) без виділення; б) вибране користувачем; в) підключене до брокера; г) відключене від брокера через помилку або втрату з'єднання

Рисунок 4.17 – Типи виділення обладнання

У таблиці 4.3 представлено вигляд об'єктів класів акторів (див. п.п. 4.2.2).

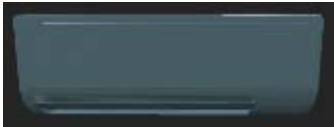
Таблиця 4.3 – Вигляд об'єктів класів акторів

Вигляд	Клас	Опис
1	2	3
	BP_FencedGround	Представляє подвір'я з парканом, реалізоване з використанням безкоштовних моделей з ресурсу TurboSquid [110]

Продовження таблиці 4.3

1	2	3
	BP_Room	Представляє кімнату, реалізована з використанням шаблонів моделей, матеріалів та текстур, які надає UE4
	BP_Door	Представляє двері, реалізовані з використанням шаблонів моделей, матеріалів та текстур, які надає UE4
	BP_Window	Представляє вікно з двома секціями, реалізоване з використанням шаблонів моделей, матеріалів та текстур, які надає UE4
	BP_Switch	Представляє вимикач, реалізований з використанням безкоштовних моделей з ресурсу TurboSquid [110]
	BP_Climate Controller	Представляє контролер системи клімат-контролю, реалізований з використанням безкоштовних моделей з ресурсу TurboSquid [110]
	BP_LightSource	Представляє звичайне джерело світла, реалізоване з використанням шаблонів моделей, матеріалів та текстур, які надає UE4

Продовження таблиці 4.3

1	2	3
	BP_RGBLight Source	Представляє RGB джерело світла, реалізоване з використанням шаблонів моделей, матеріалів та текстур, які надає UE4
	BP_Climate ControlSystem	Представляє систему клімат-контролю, реалізована з використанням безкоштовних моделей з ресурсу TurboSquid [110]
	BP_Humidity Sensor	Представляє датчик вологості, реалізований з використанням безкоштовних моделей з ресурсу TurboSquid [110]
	BP_Temperature Sensor	Представляє датчик температури, реалізований з використанням безкоштовних моделей з ресурсу TurboSquid [110]
	BP_Lightness Sensor	Представляє датчик освітленості, реалізований з використанням безкоштовних моделей з ресурсу TurboSquid [110]
	BP_Movement Sensor	Представляє датчик руху, реалізований з використанням безкоштовних моделей з ресурсу TurboSquid [110]
	BP_TopDown Character	Представляє ігрового персонажа, є стандартним класом шаблону гри «Top Down», який надає UE4

Отже, розроблений GUI ВС є зручним та привабливим. Він надає користувачу всю необхідну інформацію про свій стан (журнали подій та опублікованих MQTT-повідомлень, індикатори з'єднання обладнання) та необхідну інформацію для взаємодії з обладнанням (дані для підключення ПАК, віджети обладнання з інформацією про нього). Такий інтерфейс здатен зацікавити користувачів, а особливо студентів, завдяки його реалізації подібно до комп'ютерної гри та з використанням відомих ігрових механік. Крім цього, наявний додатковий інструмент для розробників у вигляді віджета, за допомогою якого вони можуть легко скласти у редакторі 3D-сцени будинок з окремих кімнат.

4.5 Тестування розробленої системи

Для тестування якості розроблених ПП було використано ручне тестування за принципом «білої скриньки».

Ручне тестування серверу полягає у тестуванні його REST API, тобто у надсиланні запитів до його точок входу та перевірці отриманих відповідей. Ручне тестування серверу відбувається у три етапи (рис. 4.18):

- а) розробка сценарію тесту;
- б) надсилання запитів за допомогою ПЗ Postman [111];
- в) перевірка отриманих відповідей.

Наприклад, сценарій тесту точки входу `/session/continue_session/session_id` (див. п.п. 3.3.1) потребує виконання наступних кроків:

- а) надіслати запит `/session` для початку нового сеансу;
- б) очікувати протягом 30 секунд на завершення MQTT-брокеру. Він завершує свою роботу, оскільки жодного обладнання не було підключено протягом 30 секунд від початку сеансу;
- в) надіслати запит `/session/continue_session/session_id`, у якому замість `session_id` вказати IC, отриманий у відповіді серверу на запит з кроку а;

г) перевірити, чи поле isContinued отриманої відповіді має значення «true».

Очікуваний результат тесту: поле isContinued, отримане у відповідь на запит /session/continue_session/session_id, має значення «true».

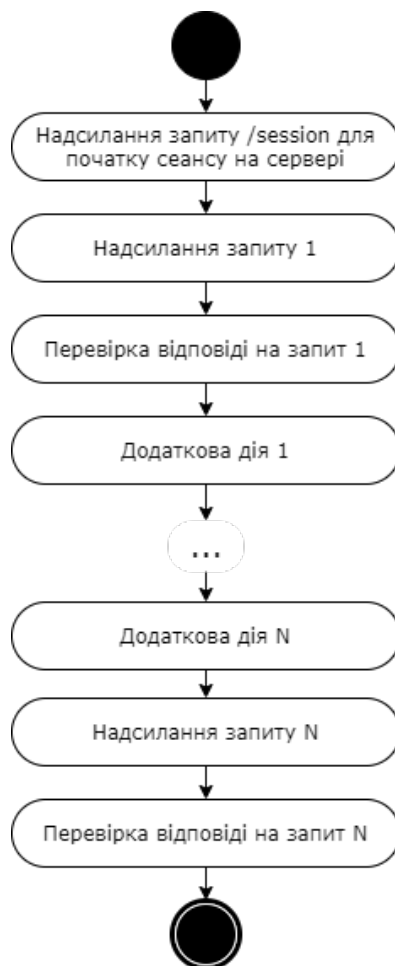


Рисунок 4.18 – Типовий сценарій тестування REST API серверу

ПЗ Postman має функцію ітераційного надсилання запитів з певним часовим інтервалом. Цю функцію можна використати для тестування механізму блокування запитів від IP у результаті перевищення ним дозволеного ліміту кількості запитів з одного IP за період часу. Для цього потрібно створити колекцію запитів та додати до неї запит /session. Після цього потрібно запустити колекцію з налаштуваннями, які представлено на рисунку 4.19, де значення Iterations має бути на один більше, ніж встановлений ліміт запитів з одного IP за період часу. Delay також потрібно встановити

такий, щоб потрібна кількість запитів встигла надійти за період часу. Вказані налаштування відповідають обмеженню в сто запитів протягом 15 хвилин [111].

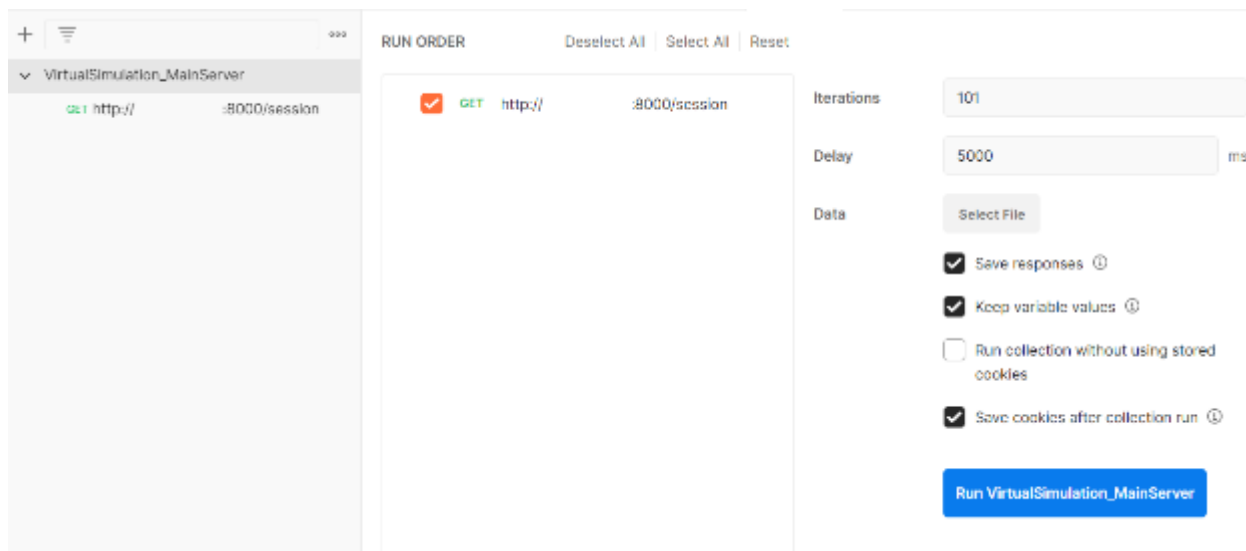


Рисунок 4.19 – Налаштування параметрів запуску колекції запитів у ПЗ Postman для тестування механізму блокування сервером запитів від IP

Очікуваний результат: на сто першій ітерації відповідь сервера має бути «429 Too Many Requests».

Отже, проведення ручного тестування за принципом «білої скриньки» за розробленими типовими сценаріями тестування як для сервера, так і для ВС, дозволяє забезпечити контроль якості розробленої системи.

4.6 Висновки до розділу 4

У даному розділі було розроблено алгоритм функціонування системи. Було описано особливості програмної реалізації серверу та ВС, виділено їх основні модулі, класи та методи. Крім цього, було описано розроблені структури даних, а також графічний інтерфейс ВС. Також було описано методику проведення тестування розробленої системи.

5 МЕТОДИКА ВИКОРИСТАННЯ РОЗРОБЛЕНОГО ОНЛАЙН ВІРТУАЛЬНОГО СЕРЕДОВИЩА РОЗУМНИЙ БУДИНОК

5.1 Умови виконання програмного продукту

Для розгортання Node.js-серверу потрібні наступні мінімальні програмно-апаратні засоби ПК:

- 4 Гб оперативної пам'яті;
- двоядерний процесор з тактовою частотою 2 ГГц;
- 35 Гб вільного місця на накопичувачі;
- ОС Ubuntu 18.04 Server;
- доменне ім'я, що вказує на публічну IP-адресу серверу;
- налаштований вебсервер Nginx SSL з використанням сертифіката

Let's Encrypt [112].

Для розгортання онлайн ВС потрібний налаштований вебхостинг з 300 Мб вільного місця на накопичувачі.

5.2 Налаштування та запуск програмного продукту

Для запуску серверу ВС потрібно виконати наступні кроки:

а) за інструкцією [112] налаштувати платформу Node.js на сервері Nginx;

б) розмістити папку з файлами розробленого серверу для ВС на ПК;

в) відкрити консоль у папці розробленого серверу;

г) послідовно виконати команди `npm install` та `npm run start_prod`.

Для налаштування ВС потрібно у папці проєкту на вебхостингу створити файл `config.ini` з наступним змістом:

- рядок `[Server]`;
- рядок `HOST=255.255.255.255`, у якому замінити значення 255.255.255.255 на хост запущеного серверу ВС;

– рядок PORT=8000, у якому замінити значення 8000 на порт запущеного серверу ВС.

5.3 Робота з програмною системою

Користувач безпосередньо взаємодіє лише з ВС.

Етап 1. Початок експерименту:

- а) перейти на вебсайт ВС;
- б) дочекатися її завантаження (рис. 5.1);
- в) ознайомитися з правилами та принципами взаємодії з ВС;
- г) натиснути кнопку «Start».

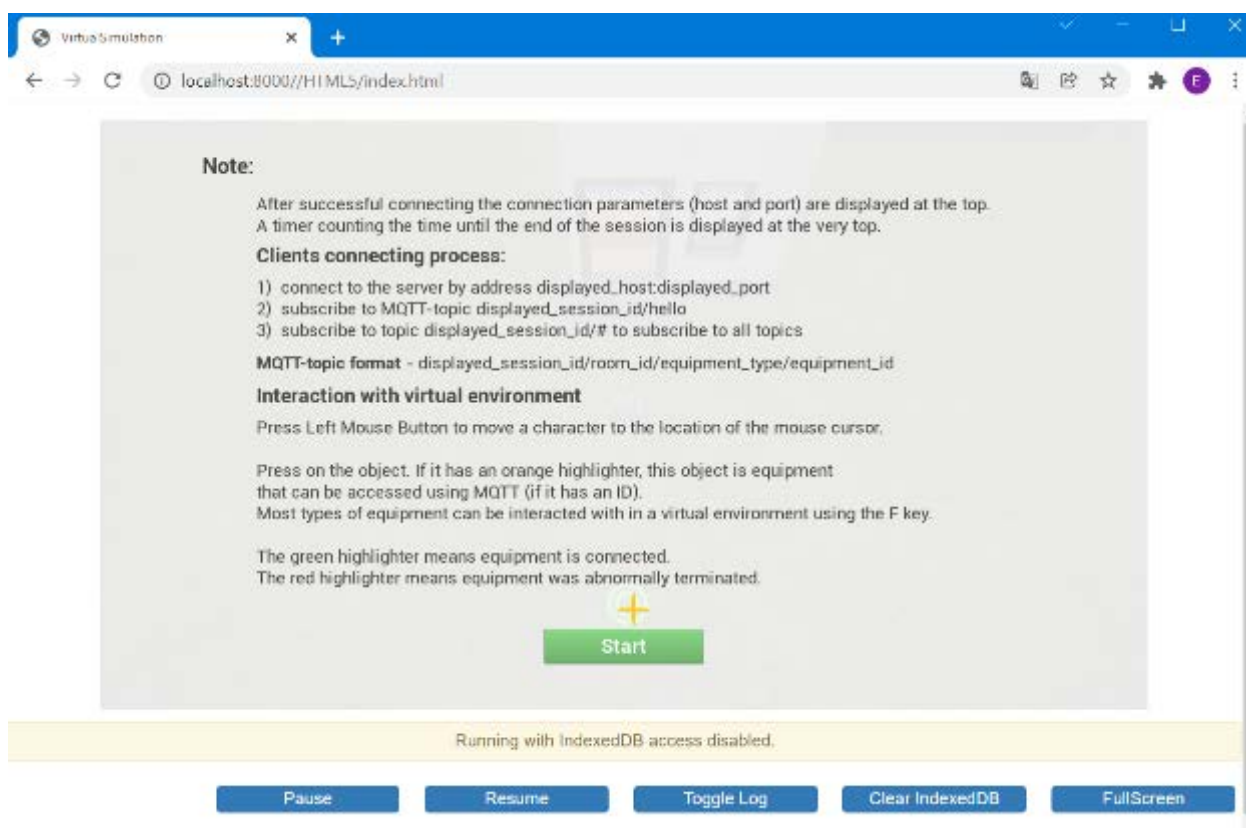


Рисунок 5.1 – Стартовий екран з правилами та принципами взаємодії з віртуальним середовищем

Етап 2. Підключення програмно-апаратних клієнтів користувача до ВС, використовуючи протокол передачі даних MQTT:

- а) ввести відображені хост, порт та ідентифікатор сеансу (рис. 5.2) у MQTT-клієнт користувача;
- б) клієнт за цими даними повинен підключитися до MQTT-брокеру ВС;
- в) клієнт повинен підписатися на тему ідентифікатор_сеансу/hello.

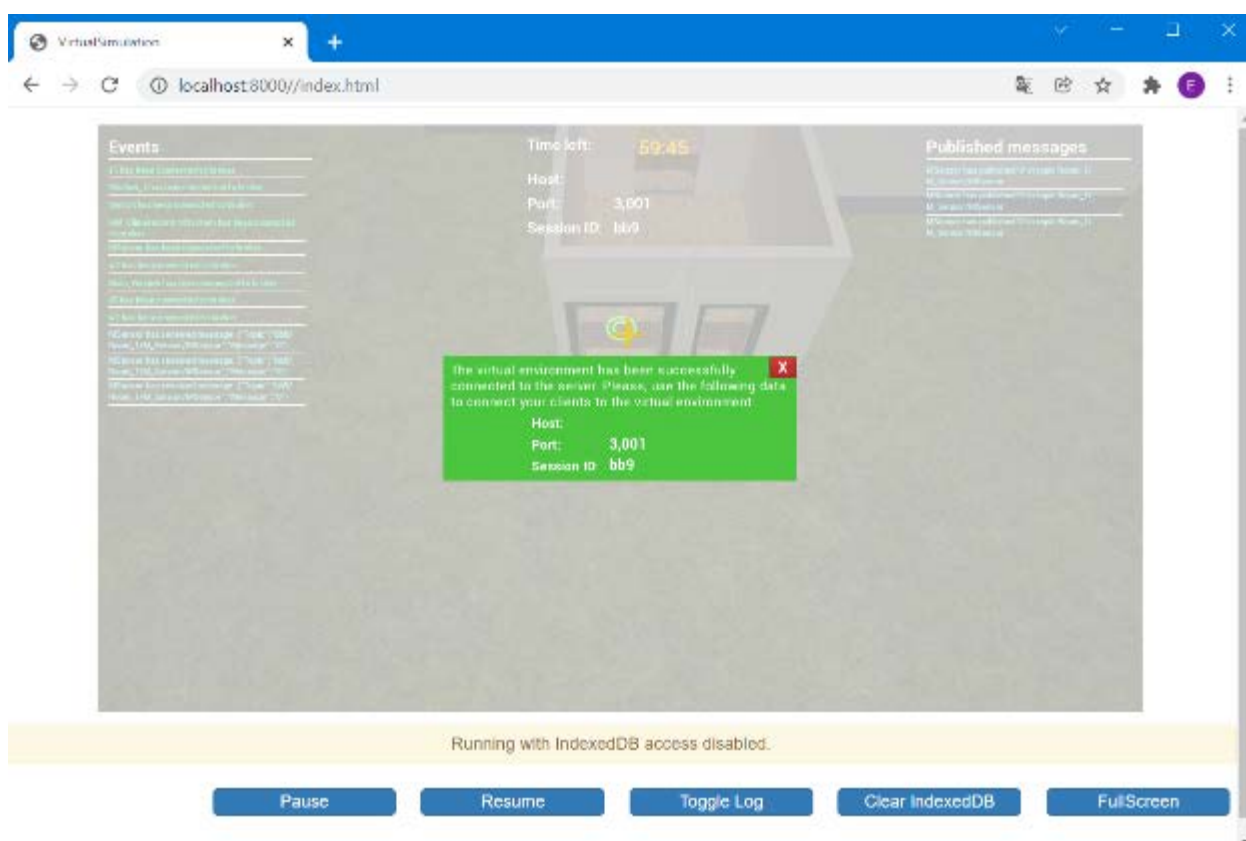


Рисунок 5.2 – Відображення даних для підключення клієнтів користувача після успішного початку експерименту

Якщо користувач не має власного клієнту, він може використати безкоштовний MQTT-клієнт, який представлено на рисунку 5.3 [113].

Для підписки одразу на усі теми обладнання потрібно підписатися на тему ідентифікатор_сеансу/#. Для підписки на повідомлення від конкретного обладнання потрібно підписатися на тему ідентифікатор_сеансу/ідентифікатор_кімнати/тип_обладнання/ідентифікатор_обладнання, де

ідентифікатор_кімнати – це синій напис на підлозі кімнати, тип_обладнання та ідентифікатор_обладнання – це дані, які відображаються у віджеті, який з’являється після клацання лівою кнопкою миші на обладнанні.



Рисунок 5.3 – Інтерфейс безкоштовного MQTT-клієнту [113]

Етап 3. Взаємодія з ВС через його графічний інтерфейс:

- а) навести курсор на потрібне місце та клацнути лівою кнопкою миші для переміщення ігрового персонажу;
- б) навести курсор на потрібне обладнання та клацнути на ньому лівою кнопкою миші для взаємодії;
- в) дочекатися, коли ігровий персонаж підійде до обладнання та зупиниться, після чого натиснути клавішу F на клавіатурі;
- г) реакція ВС на дію залежить від типу обладнання та може бути двох видів – відображення анімації зміни стану обладнання або відображення віджету для взаємодії з ним (приклад віджету наведено на рисунку 5.4).

Identificator: ClimateControlSystem
 Current temperature: 20 °C
 Current humidity: 55%

New temperature: 25 °C
 New humidity: 45%

SUBMIT Cancel

Рисунок 5.4 – Віджет для взаємодії з системою клімат-контролю

Етап 4. Взаємодія з ВС через клієнт користувача:

а) дізнатися тему, у яку обладнання публікує свої повідомлення. Це можна зробити самостійно, спираючись на ідентифікатор кімнати, ідентифікатор обладнання та тип обладнання або за допомогою переліку типів та команд обладнання, наведених у таблиці 5.1. Крім цього, можна виконати дії над обладнанням та подивитися у журналі опублікованих повідомлень, у яку тему опублікувало повідомлення це обладнання (рис. 5.5);

б) дізнатися існуючі команди для конкретного типу обладнання. Це можна також зробити експериментальним шляхом, викликаючи різні дії обладнання та переглядаючи опубліковані ним повідомлення, або за допомогою таблиці 5.1.



Рисунок 5.5 – Журнал опублікованих повідомлень, у якому можна знайти теми та команди, які використовує обладнання

Таблиця 5.1 – Типи та команди обладнання

Обладнання	Тип	Команда	Опис команди
Двері	Door	opened	Відкриває двері
		closed	Закриває двері
Вікно	Window	opened	Відкриває вікно
		closed	Закриває вікно
Вимикач світла	Switch	turn_on / turn_off	Вмикає / вимикає джерело світла, прив'язане до вимикача
		set_color/відтінок, насиченість, яскравість	Встановлює колір світла для RGB-джерела світла
Клімат контролер	Climate Controller	turn_on / turn_off	Вмикає / вимикає систему клімат-контролю, прив'язану до вимикача
		set_temperature/дійсне число	Встановлює значення температури, яку повинна підтримувати прив'язана система клімат-контролю
		set_humidity/ціле число	Встановлює значення вологості, яку повинна підтримувати прив'язана система клімат-контролю
Датчик освітленості	L_Sensor	значення	Оновлює значення освітленості у приміщенні, команда тільки для читання
Датчик руху	M_Sensor	значення	Оновлює інформацію про наявність рухомих об'єктів у приміщенні, команда тільки для читання
Датчик температури	T_Sensor	значення	Оновлює значення температури у приміщенні, команда тільки для читання
Датчик вологості	H_Sensor	значення	Оновлює значення вологості у приміщенні, команда тільки для читання

5.4 Висновки до розділу 5

У даному розділі була розроблена методика використання розробленого онлайн віртуального середовища Розумний будинок. Було наведено мінімальні програмно-апаратні засоби ПК для запуску серверу, а також вимоги до вебхостингу ВС. Було описано процес налаштування серверу та ВС.

Крім цього, було описано етапи взаємодії користувача з ВС, а саме: початок експерименту, підключення ПАК користувача до ВС, взаємодія з ВС через її GUI та взаємодія з ВС за допомогою ПАК користувача. Для кожного етапу було наведено послідовність потрібних дій. Також було перелічено типи та команди обладнання, які користувач може надсилати через власні ПАК за протоколом передачі даних MQTT.

ВИСНОВКИ

У результаті виконання роботи було досліджено віртуальні середовища та реалізовано онлайн віртуальне середовище Розумний будинок для гібридної лабораторії GOLDi. Проведені дослідження предметної галузі дозволили визначити основні існуючі технології онлайн-інженерії, а також можливості існуючих віртуальних середовищ та лабораторій. Проведені дослідження показали, що у гібридній лабораторії GOLDi реалізовано багато видів експериментів з використанням віддаленого обладнання, а також їх віртуальних моделей. Проте дані експерименти не надають можливості для проведення досліджень систем Розумний будинок та IoT технологій, хоча реалізація такого експерименту є потрібною для навчання студентів інженерних спеціальностей технологіям розробки систем домашньої автоматизації.

Для вирішення цієї проблеми було проведено дослідження технологій реалізації віртуальних середовищ. Проведені дослідження дозволили виділити особливості існуючих віртуальних середовищ та визначити методи організації взаємодії користувачів з віртуальним середовищем. На основі досліджених методів взаємодії було розроблено метод організації взаємодії користувачів з віртуальним середовищем. Крім цього, було досліджено та обрано технології для реалізації віртуального середовища для гібридної лабораторії GOLDi.

Наукова новизна отриманих результатів полягає у тому, що удосконалено метод організації взаємодії користувачів з віртуальним середовищем завдяки впровадженню методу віддаленого програмування для забезпечення нових способів отримання даних від користувача, підвищення зручності їх обробки, використання та простоти подання.

Практична цінність одержаних результатів полягає у тому, що використання розробленого онлайн віртуального середовища Розумний будинок дозволяє викладачам створювати різні реалістичні завдання, пов'язані з функціонуванням систем Розумний будинок, а студентам –

отримати досвід з проектування та розробки клієнтів для роботи з системами IoT та Розумний будинок, а також з розробки сценаріїв для домашньої автоматизації та оцінки їх ефективності.

У подальшому планується збільшити кількість доступного обладнання, додати моделі різних меблів, а також реалізувати нові можливості взаємодії ігрового персонажу з обладнанням.

Основні положення й результати роботи доповідалися й обговорювалися на Міжнародній конференції ICL2021 24th International Conference on Interactive Collaborative Learning & 50th IGIP International Conference on Engineering Pedagogy, Дрезден, Німеччина, 22-24 вересня 2021 року. Також результати роботи були представлені в тезах доповіді НПК «Тиждень науки – 2021».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kapilan, N. Virtual Laboratory: A Boon to the Mechanical Engineering Education During Covid-19 Pandemic / N. Kapilan, P. Vidhya, X.-Z. Gao // Higher Education for the Future. – 2020. – Vol. 8, Issue 1. – P. 31–46.
2. Nataro, C. A Community Springs to Action to Enable Virtual Laboratory Instruction / C. Nataro, A. R. Johnson // Journal of Chemical Education. – 2020. – Vol. 97, Issue 9. – P. 3033–3037.
3. The Application of the Remote Lab for Studying the Issues of Smart House Systems Power Efficiency, Safety and Cybersecurity / [A. Parkhomenko, A. Tulenkov, A. Sokolyanskii et al.] // International conference on Remote engineering and virtual instrumentation, Duesseldorf, Germany, 21-23 March 2018 : proceedings. – Dusseldorf : IEEE, 2018. – P. 632–639.
4. Disruptive Innovation in the Era of Global Cyber-Society: with Focus on Smart City Efforts / [A. Rucinski, R. Garbos, J. Jeffords et al.] // International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Bucharest, Romania, 21-23 September 2017 : proceedings. – Bucharest : IEEE, 2017. – P. 1102–1104.
5. How IoT, AAI Can Contribute to Smart Home and Smart Cities Services: The Role of Innovation / [K. E. Skouby, P. Lynggaard, I. Windekilde et al.] // Disruptive Innovation in the ICT Industries: Challenges for European Policy and Business : 25th European Regional Conference of the International Telecommunications Society (ITS), Brussels, Belgium, 22-25 June 2014 : proceedings. – Calgary : ITS, 2014. – P. 241–245.
6. Integrated complex for IoT technologies study / [A. Parkhomenko, A. Tulenkov, A. Sokolyanskii et al.] // Online Engineering & Internet of Things. – Cham : Springer. – 2018. – Vol. 22. – P. 322–330 (Book series Lecture Notes in Network and Systems).
7. Auer, M. E. Online Engineering and Society 4.0 / M. E. Auer, K. R. Bhimavaram, X.-G. Yue. – Cham : Springer, 2021. – 572 p.

8. Internet Technologies definition [Electronic resource]. – Access mode: <https://www.lawinsider.com/dictionary/internet-technologies>.
9. About the International Journal of Online and Biomedical Engineering (iJOE) [Electronic resource]. – Access mode: <https://online-journals.org/index.php/i-joe/about>.
10. Remote Engineering Solutions for Industrial Maintenance / [A. Kolesnikow, R. Behrens, C. Westerkamp et al.] // 11th IEEE International Conference on Industrial Informatics (INDIN), Bochum, Germany, 29-31 July 2013 : proceedings. – Piscataway, N.J. : IEEE, 2013. – P. 488–493.
11. Pinto, G. A. Remote Engineering for a Cheese Whey Biorefinery: An Internet-based Application for Process Design, Economic Analysis, Monitoring, and Control of Multiple Plant Sites / G. A. Pinto, R. L. C. Giordano, R. C. Giordano // Bioprocess and Biosystems Engineering. – 2008. – Vol. 32, Issue 1. – P. 69–78.
12. Chen, X. Virtual and Remote Laboratory Development: A Review / X. Chen, G. Song, Y. Zhang // 12th Biennial International Conference on Engineering, Construction, and Operations in Challenging Environments; and Fourth NASA/ARO/ASCE Workshop on Granular Materials in Lunar and Martian Exploration, Honolulu, Hawaii, United States, 14-17 March 2010 : proceedings. – ASCE, 2010. – P. 3843–3852.
13. Soriano, A. Virtual Laboratory to Face New Challenges in the Industry: Virtual Laboratory as Part of the New Education Age / A. Soriano, P. Ponce, A. Molina // Revolutionizing Education in the Age of AI and Machine Learning. – Hershey, Pennsylvania : IGI Global, 2019. – P. 114–129.
14. Umborg, J. Continuity in the Development of Technical Thinking / J. Umborg, A. Uukkivi // Developing Technology Mediation in Learning Environments. – Hershey, Pennsylvania : IGI Global, 2019. – P. 96–116.
15. Kennepohl, D. Remote Teaching Laboratories in Science and Engineering / D. Kennepohl // Encyclopedia of Distance Learning, Second Edition. – Hershey, Pennsylvania : IGI Global, 2009. – Vol. 3. – P. 1749–1753.
16. Lasky, V. What are Remote Laboratories? [Electronic resource] /

V. Lasky. – Access mode: <https://remotelaboratory.com/remote-laboratories/what-are-remote-laboratories/>.

17. Titov, I. Network Architectures of Remote Laboratories: Proposal of a New Solution and Comparative Analysis with Existing Ones / I. Titov, E. Smirnova // International Journal of Online and Biomedical Engineering (iJOE). – 2013. – Vol. 9, Issue 6. – P. 41–44.

18. Middleware solutions for service-oriented remote laboratories: A review / [M. Tawfik, E. Sancristobal, S. Ros et al.] // IEEE Global Engineering Education Conference (EDUCON), Istanbul, Turkey, 3-5 April 2014 : proceedings. – Istanbul : IEEE, 2014. – P. 74–82.

19. Pereira, C. E. Control and automation engineering education: Combining physical, remote and virtual labs / C. E. Pereira, S. Paladini, F. E. Schaf // 9th International Multi-Conference on Systems, Signals and Devices (SSD), Chemnitz, Germany, 20-23 March 2012 : proceedings. – Piscataway, NJ : IEEE, 2012. – P. 1–10.

20. Design considerations for virtual laboratories: A comparative study of two virtual laboratories for learning about gas solubility and colour appearance / [B. S. Wästberg, T. Eriksson, G. Karlsson et al.] // Education and Information Technologies. – 2019. – Vol. 24, Issue 3. – P. 2059–2080.

21. Darooei, R. BarBam: a new arcophobia virtual reality game / R. Darooei, A. Vard, H. Rabbani // International Serious Games Symposium (ISGS), Tehran, Iran, 26-29 December 2019 : proceedings. – Piscataway, NJ : IEEE, 2019. – P. 48–53.

22. Enhancing discipline specific skills using a virtual environment built with gaming technology / [S. Smith, K. Maund, T. Hilaire et al.] // International Journal of Work-Integrated Learning (IJWIL). – 2020. – Vol. 21, Issue 3. – P. 193–209.

23. Trenholme, D. Computer game engines for developing first-person virtual environments / D. Trenholme, S. P. Smith // Virtual Reality. – 2008. – Vol. 12, Issue 3. – P. 181–187.

24. Hanisch, A. Initialization of online simulation models / A. Hanisch, J. Tolujew, T. Schulze // Winter Simulation Conference, Orlando, FL, USA, 4 December 2005 : proceedings. – Piscataway, NJ : IEEE, 2005. – P. 1795–1803.
25. Applications of online simulation supporting PWR operations / [C. Wang, Q. Duan, C. Zhang et al.] // Nuclear Engineering and Technology. – 2021. – Vol. 53, Issue 3. – P. 842–850.
26. Chen, H. Online Modeling With Tunable RBF Network / H. Chen, Y. Gong, X. Hong // IEEE Transactions on Cybernetics. – 2013. – Vol. 43, Issue 3. – P. 935–947.
27. Towards New Multiplatform Hybrid Online Laboratory Models / [L. Rodriguez-Gil, J. Garcia-Zubia, P. Orduna et al.] // IEEE Transactions on Learning Technologies. – 2017. – Vol. 10, Issue 3. – P. 318–330.
28. Modular Web-Based Interactive Hybrid Laboratory Framework for Research and Education / [Z. Lei, H. Zhou, W. Hu et al.] // IEEE Access. – 2018. – Vol. 6. – P. 20152–20163.
29. Sumathi, S. LabVIEW based Advanced Instrumentation Systems / S. Sumathi, P. Surekha. – Berlin : Springer, 2007. – 728 p.
30. Exploring Chemistry Using Virtual Instrumentation – Challenges and Successes / [L. M. Gorghiu, G. Gorghiu, T. Alexandrescu et al.] // 5th International Conference on Multimedia and Information & Communication Technologies in Education, Lisbon, Portugal, 22-24 April 2009 : proceedings. – Badajoz : Formatex, 2009. – Vol. 1. – P. 371–375.
31. Nirubhanjali, R. V. Virtual Instrumentation / R. V. Nirubhanjali, S. Soundharya // International Journal of Engineering Research & Technology (IJERT). – 2019. – Vol. 7, Issue 11. – P. 1–2.
32. IoTIFY Home Page [Electronic resource]. – Access mode: <https://iotify.io/>.
33. Introducing IoTIFY Virtual Lab [Electronic resource]. – Access mode: <https://iotify.io/iot-virtual-lab/>.
34. IoTIFY Virtual Lab Documentation [Electronic resource]. – Access

mode: <https://docs.vlab.iotify.io/>.

35. IoTIFY IoT Simulator [Electronic resource]. – Access mode: <https://iotify.io/iot-network-simulator/>.

36. Home I/O Home Page [Electronic resource]. – Access mode: <https://realgames.co/home-io/>.

37. About Home I/O [Electronic resource]. – Access mode: <https://docs.realgames.co/homeio/en/>.

38. Virtual IOT Home page [Electronic resource]. – Access mode: <http://www.virtualiot.net/>.

39. How to Use Virtual IOT [Electronic resource]. – Access mode: <http://www.virtualiot.net/how-to-use/>.

40. OpenSHS: Open Smart Home Simulator / [N. Alshammari, T. Alshammari, M. Sedky et al.] // Sensors. – 2017. – Vol. 17, Issue 5. – P. 1– 19.

41. OpenSHS [Electronic resource]. – Access mode: <https://github.com/openshs/openshs>.

42. Wokwi Home Page [Electronic resource]. – Access mode: <https://wokwi.com/>.

43. Wokwi Documentation [Electronic resource]. – Access mode: <https://docs.wokwi.com/>.

44. UbiSim: Multiple Sensors Mounted Smart House Simulator Development / [W. Lee, S. Cho, W. Song et al.] // 11th International Conference on Dependable, Autonomic and Secure Computing, Chengdu, China, 21-22 December 2013 : proceedings. – Chengdu : IEEE, 2013. – P. 450–453.

45. Kucera, E. Connection between 3D engine unity and microcontroller arduino: A virtual smart house / E. Kucera, O. Haffner, S. Kozak // Cybernetics & Informatics (K&I), Lazy pod Makytou, Slovakia, 31 January-3 February 2018 : proceedings. – Piscataway, NJ : IEEE, 2018. – P. 1–8.

46. Mîndruț, B. M. Experiential Learning Through Controlling and Monitoring a Real-Time 3D House Using LabVIEW in a Virtual Laboratory / B. M. Mîndruț, C. A. Oprea // Innovative Technologies and Learning. – Cham :

Springer. – 2020. – Vol. 12555. – P. 119–127 (Book series Lecture Notes in Computer Science).

47. GOLDi – Grid of Online Lab Devices Ilmenau / [K. Henke, T. Vietzke, H.-D. Wuttke et al.] // International Journal of Online Engineering (iJOE). – 2016. – Vol. 12, Issue 4. – P. 11–13.

48. Presentation of the GOLDi lab architecture [Electronic resource]. – Access mode: <https://www.goldi-labs.net/index.php?Site=2>.

49. The Grid of Online Laboratory Devices Ilmenau (GOLDi) welcomes you [Electronic resource]. – Access mode: <https://www.goldi-labs.net/index.php>.

50. How to use ECP [Electronic resource]. – Access mode: <https://www.goldi-labs.net/index.php?Site=70>.

51. Preparing to the experiment [Electronic resource]. – Access mode: <https://www.goldi-labs.net/index.php?Site=5>.

52. Desktop VR Is Better Than Non-ambulatory HMD VR for Spatial Learning / [P. Srivastava, A. Rimzhim, P. Vijay et al.] // Frontiers in Robotics and AI. – 2019. – Vol. 6. – P. 1–15.

53. Desktop versus immersive virtual environments: effects on spatial learning / [J. Zhao, T. Sensibaugh, B. Bodenheimer et al.] // Spatial Cognition & Computation. – 2020. – Vol. 20, Issue 4. – P. 328–363.

54. Remote Lab Experiments Models: A Comparative Study / [I. Mougharbel, A. E. Hajj, H. Artail et al.] // International Journal of Engineering Education (IJEE). – 2006. – Vol. 22, Issue 4. – P. 849–857.

55. Carroll, J. M. Human Computer Interaction – brief intro [Electronic resource] / J. M. Carroll. – Access mode: <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro>.

56. Hinckley, K. Input Technologies and Techniques / K. Hinckley, D. Wigdor // The Human-Computer Interaction Handbook, Third Edition. – Boca Raton, FL : CRC Press, 2012. – P. 95–132.

57. Inkpen, K. M. Drag-and-Drop versus Point-and-Click Mouse

Interaction Styles for Children / K. M. Inkpen // ACM Transactions on Computer-Human Interaction. – 2001. – Vol. 8, Issue 1. – P. 1–33.

58. Selection [Electronic resource]. – Access mode: <https://material.io/design/interaction/selection.html>.

59. Kaley, A. Contextual Menus: Delivering Relevant Tools for Tasks [Electronic resource] / A. Kaley. – Access mode: <https://www.nngroup.com/articles/contextual-menus/>.

60. Caroux, L. Influence of head-up displays' characteristics on user experience in video games / L. Caroux, K. Isbister // International Journal of Human-Computer Studies. – 2016. – Vol. 87. – P. 65–79.

61. Player–video game interaction: A systematic review of current concepts / [L. Caroux, K. Isbister, L. Le Bigot et al.] // Computers in Human Behavior. – 2015. – Vol. 48. – P. 366–381.

62. Garcia, J. Perspectives and Points of View [Electronic resource] / J. Garcia. – Access mode: https://www.ign.com/wikis/video-game-dictionary/Perspectives_and_Points_of_View.

63. Compton, C. Run, Jump and Climb: Designing Fun Movement in Games [Electronic resource] / C. Compton. – Access mode: <https://www.gamedeveloper.com/audio/run-jump-and-climb-designing-fun-movement-in-games>.

64. Caicedo, D. G. What are tangible user interfaces? [Electronic resource] / D. G. Caicedo. – Access mode: <https://medium.com/@bluehair.co/what-are-tangible-user-interfaces-ce6a5aa5c912>.

65. A multimodal interface to control a robot arm via the web: a case study on remote programming / [R. Marin, P. J. Sanz, P. Nebot et al.] // IEEE Transactions on Industrial Electronics. – 2005. – Vol. 52, Issue 6. – P. 1506–1520.

66. MATLAB Overview [Electronic resource]. – Access mode: <https://www.mathworks.com/products/matlab.html>.

67. Hinov, N. A Web-Based Matlab Application for E-Learning and Simulation of Electric Vehicle Performance / N. Hinov, H. Kanchev, F. Krastev //

7th International Conference on Energy Efficiency and Agricultural Engineering (EE&AE), Ruse, Bulgaria, 12-14 November 2020 : proceedings. – Ruse : IEEE, 2020. – P. 1–4.

68. MATLAB Simulink Overview [Electronic resource]. – Access mode: <https://www.mathworks.com/products/simulink.html>.

69. MATLAB Simulink 3D Animation Overview [Electronic resource]. – Access mode: <https://www.mathworks.com/products/3d-animation.html>.

70. Simulink 3D Animation Web Viewer Documentation [Electronic resource]. – Access mode: <https://www.mathworks.com/help/sl3d/when-to-use-the-web-viewer.html>.

71. Get Started with MATLAB Web App Server [Electronic resource]. – Access mode: <https://www.mathworks.com/help/webappserver/getting-started-with-matlab-web-app-server.html>.

72. Bistak, P. 3D Three-Tank Remote Laboratory Based on Matlab and Websockets / P. Bistak // 5th Experiment International Conference, Funchal, Portugal, 12-14 June 2019 : proceedings. – Piscataway, NJ : IEEE, 2019. – P. 85–89.

73. Bedillion, M. Virtual Laboratories Using Simulink: A Pilot Study / M. Bedillion, M. H. Mohamed Nizar // 122nd Annual Conference & Exposition, Seattle, USA, 14-17 June 2015 : proceedings. – Seattle, WA : ASEE, 2015. – P. 1–12.

74. Sinicki, A. What is Unity? Everything you need to know [Electronic resource] / A. Sinicki. – Access mode: <https://www.androidauthority.com/what-is-unity-1131558/>.

75. Gregory, J. Game Engine Architecture, Third Edition / J. Gregory. – Boca Raton, FL : A K Peters/CRC Press, 2018. – 1240 p.

76. Manning, J. Mobile game development with Unity / J. Manning, P. Buttfield-Addison. – Sebastopol, CA : O'Reilly Media, 2017. – 464 p.

77. Plans and pricing [Electronic resource]. – Access mode: <https://store.unity.com/>.

78. How to publish for WebGL [Electronic resource]. – Access mode: <https://learn.unity.com/tutorial/how-to-publish-for-webgl#>.
79. Design of a collaborative virtual environment for training security agents in big events / [C. Passos, M. H. da Silva, A. C. A. Mol et al.] // Cognition, Technology & Work. – 2017. – Vol. 19. – P. 315–328.
80. Gestwicki, P. Unreal Engine 4 for Computer Scientists / P. Gestwicki // Journal of Computing Sciences in Colleges. – 2019. – Vol. 35, Issue 5. – P. 109–110.
81. Sheverev, D. A. The development of a virtual laboratory based on Unreal Engine 4 / D. A. Sheverev, I. N. Kozlova // Сборник трудов IV международной конференции и молодежной школы «Информационные технологии и нанотехнологии» (ИТНТ-2018). – Самара : Новая техника, 2018. – С. 2248–2252.
82. Carnall, B. Unreal Engine 4.X by Example / B. Carnall. – Birmingham, UK : Packt Publishing, 2016. – 506 p.
83. Unreal Engine licensing options [Electronic resource]. – Access mode: <https://www.unrealengine.com/en-US/download>.
84. HTML5 SDK Setup [Electronic resource]. – Access mode: <https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/HTML5/GettingStarted>.
85. Emscripten [Electronic resource]. – Access mode: <https://emscripten.org/>.
86. A 3D virtual Weft-knitting Engineering learning system based on Unreal Engine 4 / [X. Zhang, J. Liu, Q. Chen et al.] // Computer Applications in Engineering Education. – 2018. – Vol. 26, Issue 6. – P. 2223–2236.
87. Babor, S. HTML5/WebGL vs Flash in 3D Visualisation / S. Babor, B. Ramic-Brkic // The 17th Central European Seminar on Computer Graphics (CESCG 2013), Smolenice, Slovakia, 28-30 April 2013 : proceedings. – Smolenice : CESCG, 2013. – P. 1–8.
88. 20 Interactive 3D Javascript Libraries & Frameworks [Electronic

resource]. – Access mode: <https://bashooka.com/coding/3d-javascript-libraries/>.

89. Paul, R. Kataspace: browser-based virtual worlds built with WebGL and HTML5 [Electronic resource] / R. Paul. – Access mode: <https://arstechnica.com/information-technology/2010/12/kataspace-browser-based-virtual-world-built-with-webgl-and-html5/>.

90. Kawamoto, L. T. Development and Validation of a Virtual Environment to Evaluate the Capacity in Executing the Safety Procedures in Recreational Scuba Diving / L. T. Kawamoto, A. F. F. Slaets, S. C. M. Rodrigues // Games for Health Journal. – 2012. – Vol. 1, Issue 2. – P. 104–114.

91. Maiti, A. Flight simulation in 3D Web Map Services using Silverlight 3 to Reduce Simulation Costs in Pilot Training Institutes / A. Maiti // The World Congress on Engineering and Computer Science (WCECS), San Francisco, USA, 20-22 October 2009 : proceedings. – Hong Kong : Newswood Limited, 2009. – Vol. 1. – P. 1–4.

92. Eldad, A. Unity vs Unreal, What Kind of Game Dev Are You? [Electronic resource] / A. Eldad. – Access mode: <https://www.incredibuild.com/blog/unity-vs-unreal-what-kind-of-game-dev-are-you>.

93. Sharma, C. Communication Protocol Stack for Constrained IoT Systems / C. Sharma, N. K. Gondhi // 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), Bhimtal, India, 23-24 February 2018 : proceedings. – Piscataway, NJ : IEEE, 2018. – P. 1–6.

94. Tukade, T. M. Data Transfer Protocols in IoT-An Overview / T. M. Tukade, R. Banakar // International Journal of Pure and Applied Mathematics. – 2018. – Vol. 118. – P. 121–138.

95. McEwen, S. Requirements: An introduction [Electronic resource] / S. McEwen. – Access mode: <http://www.uml.org.cn/requirementproject/200702131.asp>.

96. Yigal, A. Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases [Electronic resource] / A. Yigal. – Access mode:

<https://logz.io/blog/relational-database-comparison/>.

97. What Is a REST API? [Electronic resource]. – Access mode: <https://www.akana.com/blog/what-is-rest-api>.

98. Introducing JSON [Electronic resource]. – Access mode: <https://www.json.org/json-en.html>.

99. Cope, S. Understanding MQTT Topics [Electronic resource] / S. Cope. – Access mode: <http://www.steves-internet-guide.com/understanding-mqtt-topics>.

100. Clark, J. Top 10 Backend Technologies [Electronic resource] / J. Clark. – Access mode: <https://blog.back4app.com/backend-technologies/>.

101. Yushkevych, A. Backend Development: How to Choose the Right Framework [Electronic resource] / A. Yushkevych. – Access mode: <https://monovm.com/blog/backend-development-how-to-choose-the-right-framework/>.

102. Clark, J. Top 10 PHP Frameworks [Electronic resource] / J. Clark. – Access mode: <https://blog.back4app.com/php-framework/>.

103. Sauve, P. Comparing JavaScript and Node.js IDEs in 2021 [Electronic resource] / P. Sauve. – Access mode: <https://neowire.co/comparing-nodejs-javascript-ides/>.

104. Wadhwani, P. 17 Top Node Js IDEs For Modern Application Development [Electronic resource] / P. Wadhwani. – Access mode: <https://www.bacancytechnology.com/blog/top-node-js-ides>.

105. Helmet [Electronic resource]. – Access mode: <https://www.npmjs.com/package/helmet>.

106. Express Rate Limit [Electronic resource]. – Access mode: <https://www.npmjs.com/package/express-rate-limit>.

107. Uuid [Electronic resource]. – Access mode: <https://www.npmjs.com/package/uuid>.

108. Aedes [Electronic resource]. – Access mode: <https://www.npmjs.com/package/aedes>.

109. Смоделируйте систему отопления дома [Электрон. ресурс]. –

Режим доступа: <https://docs.exponenta.ru/simulink/ug/model-a-house-heating-system.html>.

110. TurboSquid [Electronic resource]. – Access mode: <https://www.turbosquid.com/>.

111. Postman [Electronic resource]. – Access mode: <https://www.postman.com/>.

112. Bearnes, B. How To Set Up a Node.js Application for Production on Ubuntu 18.04 [Electronic resource] / B. Bearnes, K. Juell. – Access mode: <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-18-04>.

113. BP_MQTT_Demo [Electronic resource]. – Access mode: https://github.com/damody/BP_MQTT_Demo/releases.

ДОДАТОК А
Текст програми

A.1 Модуль sessionRouter

```
import { Response, Request, Router } from 'express';

import { IFinishSessionSuccessResponse } from
'../services/SessionsService/types';
import { SessionsServiceInstance } from '../services';
import { errorResponseGenerator, sessionActionsHandler } from '../utilities';

const session = async (_, Request, res: Response) => {
  try {
    const session = await SessionsServiceInstance.startSession();
    res.status(200).json({ session });
  } catch (err) {
    res.status(500).json(errorResponseGenerator(err.toString()));
  }
};

const reconnect = async (req: Request, res: Response) => {
  await sessionActionsHandler(
    req,
    res,
    async () => await SessionsServiceInstance.reconnect(req.params.session_id),
    'session',
  );
};

const startSessionTimer = async (req: Request, res: Response) => {
  await sessionActionsHandler(
    req,
    res,
    async () => SessionsServiceInstance.startSessionTimer(req.params.session_id),
    'response',
  );
};

const finishSession = async (req: Request, res: Response) => {
  await sessionActionsHandler(
    req,
    res,
    async () => {
      const session_id = req.params.session_id;
      SessionsServiceInstance.closeSession(session_id);
      const response: IFinishSessionSuccessResponse = {
        session_id,
        isFinished: true,
      };
      return response;
    },
  );
};
```

```

        'response',
    );
};

const continueSession = async (req: Request, res: Response) => {
    await sessionActionsHandler(
        req,
        res,
        async () => SessionsServiceInstance.continueSession(req.params.session_id),
        'response',
    );
};

export const sessionRouter = Router();

sessionRouter.get('/', session);
sessionRouter.post('/continue_session/:session_id', continueSession);
sessionRouter.post('/finish_session/:session_id', finishSession);
sessionRouter.post('/reconnect/:session_id', reconnect);
sessionRouter.post('/start_session_timer/:session_id', startSessionTimer);

```

A.2 Клас BrokerService

```

import aedes, { Aedes, Client as AedesClient, Subscription } from 'aedes';
import { AddressInfo } from 'net';
import { Server as Broker, createServer } from 'aedes-server-factory';
import { printf } from 'fast-printf';

import { CLIENT_DISCONNECTED, VIOLATES_HANDSHAKE_PROCEDURE } from
'../../constants';
import { IBrokerAddressInfo } from './types';
import { LogFunction } from '../../types';
import {
    brokerErrorLog,
    brokerInfoLog,
    brokerSuccessLog,
    helloTopic,
    successLog,
} from '../../utilities';

// 'HELLO'-message, <session ID>/hello
// subscribe on $SYS-topic - $SYS/<aedes_id(in our case -
session_id)>/<topic_name(for example, heartbeat)>

export class BrokerService {
    // Broker related fields
    private broker!: Aedes;
    private brokerServer!: Broker;
    // Clients related fields

```

```

private clients: Map<string, AedesClient> = new Map();
private clientsIsHelloSent: Map<string, boolean> = new Map();
// Connection parameters
private session_id: string;
private host: string;
private clientsPort: number;
// Log functions
private ownErrorLog: LogFunction;
private ownInfoLog: LogFunction;
private ownSuccessLog: LogFunction;

/*****
 * Public methods *
*****/

constructor(session_id: string, host: string, clientsPort: number) {
  this.session_id = session_id;
  this.host = host;
  this.clientsPort = clientsPort;

  this.initBroker();

  this.ownErrorLog = (errorMessage) => brokerErrorLog(session_id,
errorMessage);
  this.ownInfoLog = (infoMessage) => brokerInfoLog(session_id, infoMessage);
  this.ownSuccessLog = (successMessage) => brokerSuccessLog(session_id,
successMessage);
}

address(): IBrokerAddressInfo {
  const { address, family } = this.brokerServer.address() as AddressInfo;
  return {
    address,
    family,
    host: this.host,
    clientsPort: this.clientsPort
  };
}

close(callback?: (err?: Error) => void): void {
  return this.broker.close(() => {
    this.brokerServer.close(() => {
      callback?.();
    });
  });
}

closeClient(id: string): void {
  const client = this.clients.get(id);
  if (client) {
    console.log('client');
  }
}

```

```

        this.clients.delete(id);
        client.close();
    }

    this.clientsIsHelloSent.delete(id);
}

/*****
 * Broker related initializer and handlers *
*****/

private initBroker(): void {
    this.broker = aedes({ id: this.session_id });
    this.brokerServer = createServer(this.broker);

    this.brokerServer.listen(this.clientsPort, () => {
        console.log('\n');
        successLog(
            `MQTT-broker for session_id ${this.session_id} has been started on port
${this.clientsPort}`,
        );
    });

    this.broker.on('clientReady', this._onClientReady);
    this.broker.on('subscribe', this._onClientSubscribe);
    this.broker.on('clientDisconnect', this._onClientDisconnect);
    this.broker.on('clientError', this._onClientError);
}

private _onClientReady = (client: AedesClient) => {
    this.ownInfoLog(`Client ${client.id} connected`);
    this.clients.set(client.id, client);
    this.clientsIsHelloSent.set(client.id, false);
};

private _onClientSubscribe = (subscriptions: Subscription[], client:
AedesClient) => {
    if (!this.clients.get(client.id)) {
        return;
    }
    const curHelloTopic = helloTopic(this.session_id);
    if (subscriptions[0].topic !== curHelloTopic &&
!this.clientsIsHelloSent.get(client.id)) {
        const error = printf(VIOLATES_HANDSHAKE_PROCEDURE, client.id);
        this.ownErrorLog(error);
        this.closeClient(client.id);
    } else if (subscriptions[0].topic === curHelloTopic) {
        this.clientsIsHelloSent.set(client.id, true);
        this.ownSuccessLog(`Client ${client.id} has been connected and
authorized`);
    }
}

```

```

};

private _onClientDisconnect = (client: AedesClient) => {
  const error = printf(CLIENT_DISCONNECTED, client.id);
  this.ownInfoLog(error);
  this.closeClient(client.id);
};

private _onClientError = (client: AedesClient, error: Error) => {
  this.ownErrorLog(`Client ${client.id} error: ${error.toString()}.`);
  this.closeClient(client.id);
};
}

```

A.3 Класс SessionsService

```

import getPort, { makeRange } from 'get-port';
import { check } from 'tcp-port-used';
import { v4 as uuidv4 } from 'uuid';

import {
  BROKER_MAX_PORT,
  BROKER_MIN_PORT,
  HOST,
  INTERNAL_ERROR_MES,
  NO_SESSION_FOUND_MES,
  SESSION_ID_LENGTH,
  SESSION_TIMELIFE,
  SESSION_TIMELIFE_EXTRA_TIME,
  SESSIONS_LIMIT,
  SESSIONS_LIMIT_MES,
} from '../constants';
import { BrokerService } from '../BrokerService';
import {
  CustomError,
  IContinueSessionSuccessResponse,
  ISessionSuccessResponse,
  IStartTimerSuccessResponse,
  SessionsLimitError,
} from './types';
import { DBServiceInstance, ISession } from '../DBService';
import { errorLog, infoLog, successLog } from '../utilities';

class SessionsService {
  private sessions: Map<string, BrokerService> = new Map();
  private timers: Map<string, NodeJS.Timeout> = new Map();
  private timestamps: Map<string, number> = new Map();

  async loadAllSessions() {

```

```

// eslint-disable-next-line no-useless-catch
try {
  const loadSessions = await DBServiceInstance.loadAllSessions();
  successLog(`Sessions have been loaded. Number of sessions:
${loadSessions.length}`);
  // timeout is needed to prevent possible errors in startSession -> getPort
  loadSessions.forEach(async (item, index) => {
    const { session_id, host, clientsPort } = item;

    if (index + 1 <= SESSIONS_LIMIT) {
      setTimeout(async () => {
        try {
          await this.startSession(session_id, host, clientsPort);
        } catch (err) {
          if (err instanceof SessionsLimitError) {
            try {
              await DBServiceInstance.removeSession(session_id);
            } catch (err) {}
          }
        }
      }, 500);
    } else {
      try {
        await DBServiceInstance.removeSession(session_id);
      } catch (err) {}
    }
  });
} catch (err) {
  throw err;
}
}

// host parameter is not used for now
async startSession(session_id?: string, host: string = HOST, clientsPort?:
number, ) {
  if (this.sessions.size >= SESSIONS_LIMIT) {
    infoLog(`startSession error: ${SESSIONS_LIMIT_MES}`);
    throw new SessionsLimitError();
  }

  let isSessionIdWasGenerated = false;

  if (!session_id) {
    try {
      let isUnique = true;
      do {
        session_id = uuidv4().substr(0, SESSION_ID_LENGTH);
        isUnique = !(await
DBServiceInstance.findSessionById(session_id)).length;
      } while (!isUnique);
    }
  }
}

```

```

        isSessionIdWasGenerated = true;
    } catch (err) {
        errorLog(`startSession generating session_id error: ${err}`);
        throw new CustomError(INTERNAL_ERROR_MES);
    }
}

try {
    clientsPort = await this.findFreeUniquePort(clientsPort);

    const broker = new BrokerService(session_id, host, clientsPort);

    if (isSessionIdWasGenerated) {
        await DBServiceInstance.saveSession(session_id, host, clientsPort);
    } else {
        try {
            await DBServiceInstance.updateSession(session_id, { host, clientsPort
        });
        } catch (err) {
            errorLog(`startSession updateSession error: ${err}`);
        }
    }

    this.sessions.set(session_id, broker);

    const newSession: ISessionSuccessResponse = {
        host,
        clientsPort,
        session_id,
        timeLeft: SESSION_TIMELIFE,
        timeToDecide: SESSION_TIMELIFE_EXTRA_TIME,
    };

    return newSession;
} catch (err) {
    errorLog(`startSession starting MQTT-broker error: ${err}`);
    /**
     * We use isSessionIdWasGenerated to prevent removing sessions, which were
loaded
     * on server start. Imagine situation when some amount of sessions were
startes,
     * but the server crashed. We restart the server, it try to load all
unfinished sessions,
     * but some error occurs. If we don't use this variable, the loaded on
startup session,
     * on which the error was received, will be removed from DB.
     */
    if (isSessionIdWasGenerated) {
        try {
            await DBServiceInstance.removeSession(session_id);
        } catch (err) {}
    }
}

```

```

    }

    throw new CustomError(INTERNAL_ERROR_MES);
  }
}

async reconnect(session_id: string) {
  if (this.sessions.has(session_id)) {
    const broker = this.sessions.get(session_id);

    let timeLeft = this.timestamps.get(session_id);
    if (timeLeft) {
      timeLeft = SESSION_TIMELIFE - (Date.now() - timeLeft);
    } else {
      timeLeft = SESSION_TIMELIFE;
    }

    if (broker) {
      const { host, clientsPort } = broker.address();

      const session: ISessionSuccessResponse = {
        host,
        clientsPort,
        session_id,
        timeLeft,
        timeToDecide: SESSION_TIMELIFE_EXTRA_TIME,
      };

      return session;
    } else throw new CustomError(INTERNAL_ERROR_MES);
  }

  try {
    const session = await DBServiceInstance.findSessionById(session_id);

    if (session.length) {
      const { host, clientsPort } = session[0] as unknown as ISession;

      const sessionsNumber = (await
DBServiceInstance.loadAllSessions()).length;
      if (sessionsNumber < SESSIONS_LIMIT) {
        return await this.startSession(session_id, host, clientsPort);
      } else {
        throw new SessionsLimitError();
      }
    } else {
      throw new CustomError(NO_SESSION_FOUND_MES);
    }
  } catch (err) {
    errorLog(`reconnect error: ${err}`);
  }
}

```

```

        if (err instanceof SessionsLimitError || err instanceof CustomError) {
            throw err;
        } else throw new CustomError(INTERNAL_ERROR_MES);
    }
}

async closeAllSessions(needToRemoveFromDB = true) {
    this.sessions.forEach((_, session_id) =>
        // Set needToRemoveFromDB to false to prevent calling remove operation on
        DB for each item
        this.closeSession(session_id, false),
    );

    if (needToRemoveFromDB) {
        try {
            // And just clean whole table
            await DBServiceInstance.cleanSessions();
        } catch (err) {
            errorLog(`closeAllSessions error: ${err}`);
        }
    }
}

startSessionTimer(session_id: string) {
    if (this.sessions.has(session_id)) {
        if (this.timers.has(session_id)) {
            const response: IStartTimerSuccessResponse = {
                session_id,
                isStarted: true,
            };
            return response;
        }

        // 2 extra minutes to have an interval between server & ve timers (ve timer
        finishes first in this case)
        const timer = setTimeout(
            () => this.closeSession(session_id),
            SESSION_TIMELIFE + SESSION_TIMELIFE_EXTRA_TIME + 120000,
        );

        successLog(`Timer for session_id ${session_id} has been started`);

        this.timers.set(session_id, timer);
        this.timestamps.set(session_id, Date.now());

        const response: IStartTimerSuccessResponse = {
            session_id,
            isStarted: true,
        };

        return response;
    }
}

```

```

    } else {
        throw new CustomError(NO_SESSION_FOUND_MES);
    }
}

continueSession(session_id: string) {
    const timer = this.timers.get(session_id);
    if (timer) {
        clearTimeout(timer);
        this.timers.delete(session_id);
    }

    const startedTimer = this.startSessionTimer(session_id);
    const response: IContinueSessionSuccessResponse = {
        session_id,
        isContinued: startedTimer.isStarted,
        timeLeft: SESSION_TIMELIFE,
    };
    return response;
}

closeSession = (session_id: string, needToRemoveFromDB = true) => {
    const broker = this.sessions.get(session_id);
    if (!broker) {
        return;
    }

    try {
        broker.close(async () => {
            if (needToRemoveFromDB) {
                try {
                    await DBServiceInstance.removeSession(session_id);
                } catch (err) {
                    errorLog(`closeSession error: ${err}`);
                }
            }
        });
    } catch (err) {
        errorLog(`closeSession error: ${err}`);
    } finally {
        this.sessions.delete(session_id);

        const timer = this.timers.get(session_id);
        if (timer) {
            console.log('here');
            clearTimeout(timer);
        }

        this.timers.delete(session_id);
        this.timestamps.delete(session_id);
    }
}

```

```

        infoLog(`Session ${session_id} will be closed`);
    }
};

private async findFreeUniquePort(port?: number | undefined) {
    let isPortFree = false;
    let newPort = port;
    if (newPort) {
        isPortFree = !(await check(newPort as number));
    }

    if (!newPort || !isPortFree) {
        let isUnique = true;
        do {
            newPort = await getPort({
                port: makeRange(BROKER_MIN_PORT, BROKER_MAX_PORT),
            });

            isUnique =
                !(await DBServiceInstance.findSessionByClientsPort(newPort)).length &&
                !(await DBServiceInstance.findSessionByVePort(newPort)).length;
        } while (!isUnique);
    }

    return newPort;
}

export const SessionsServiceInstance = new SessionsService();

```

A.4 Класс DBService

```

import { knex } from 'knex';

// @ts-ignore
import config from '../../../knexfile';
import { IUpdateSession } from './types';
import { errorLog, infoLog } from '../../../utilities';

export * from './types';

class DBService {
    private db = knex(config.development);

    saveSession(session_id: string, host: string, clientsPort: number) {
        return new Promise<number[]>((resolve, reject) => {
            this.db('sessions')
                .insert({ session_id, host, clientsPort })
                .then((value) => {

```

```

        infoLog(`session saved: session_id = ${session_id}`);
        resolve(value);
    })
    .catch((err) => {
        errorLog(`saveSession error: ${err}`);
        reject(err);
    });
});
}

findSessionById(session_id: string) {
    return this.db('sessions').where({ session_id });
}

findSessionByClientsPort(clientsPort: number) {
    return this.db('sessions').where({ clientsPort });
}

updateSession(session_id: string, newValues: IUpdateSession) {
    return new Promise<number>((resolve, reject) => {
        const { host, clientsPort } = newValues;
        this.db('sessions')
            .where({ session_id })
            .update({
                host,
                clientsPort
            })
            .then((value) => {
                infoLog(`session updated: session_id = ${session_id}`);
                resolve(value);
            })
            .catch((err) => {
                errorLog(`updateSession error: ${err}`);
                reject(err);
            });
    });
}

removeSession(session_id: string) {
    return new Promise<number>((resolve, reject) => {
        this.db('sessions')
            .where({ session_id })
            .del()
            .then((value) => {
                infoLog(`removed session_id: ${session_id}`);
                resolve(value);
            })
            .catch((err) => {
                errorLog(`removeSessions error: ${err}`);
                reject(err);
            });
    });
}

```

```
    });  
  }  
  
  loadAllSessions() {  
    return this.db('sessions');  
  }  
  
  cleanSessions() {  
    return this.db('sessions').truncate();  
  }  
}  
  
export const DBServiceInstance = new DBService();
```

ДОДАТОК Б
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра

**Дослідження та розробка віртуального середовища
для гібридної лабораторії GOLDi**

Виконав
ст. гр. КНТ-120м

Є. В. Яремченко

Керівник
к.т.н., доцент

А. В. Пархоменко

2021

Рисунок Б.1 – Титульний слайд



Рисунок Б.2 – Актуальність роботи

Мета та задачі

Мета – дослідження та програмна реалізація віртуального онлайн-середовища Розумний будинок для розширення функціональних можливостей гібридної лабораторії GOLDi.

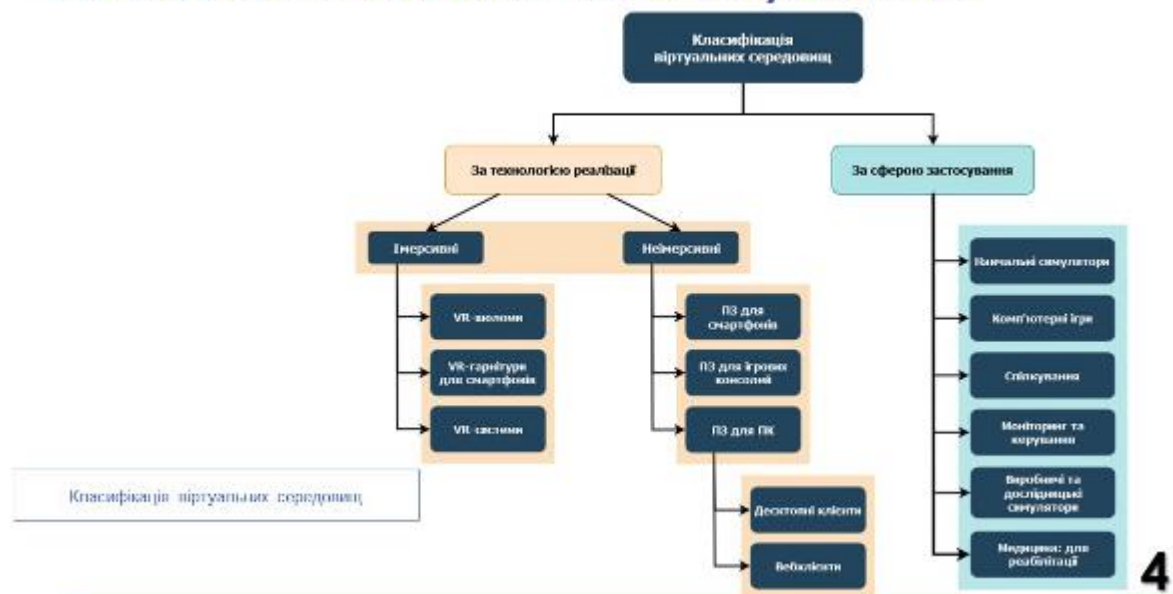
Задачі:

- дослідження особливостей існуючих ВС;
- дослідження та розробка методу організації взаємодії користувачів з ВС;
- аналіз технологій реалізації ВС;
- аналіз протоколів передачі даних, що використовуються в системах Розумний будинок та IoT;
- програмна реалізація онлайн ВС Розумний будинок та його тестування;
- розробка методики використання створеного онлайн ВС Розумний будинок.

3

Рисунок Б.3 – Мета та задачі

Дослідження особливостей існуючих ВС



4

Рисунок Б.4 – Дослідження особливостей існуючих ВС

Дослідження методів організації взаємодії користувачів з віртуальним середовищем

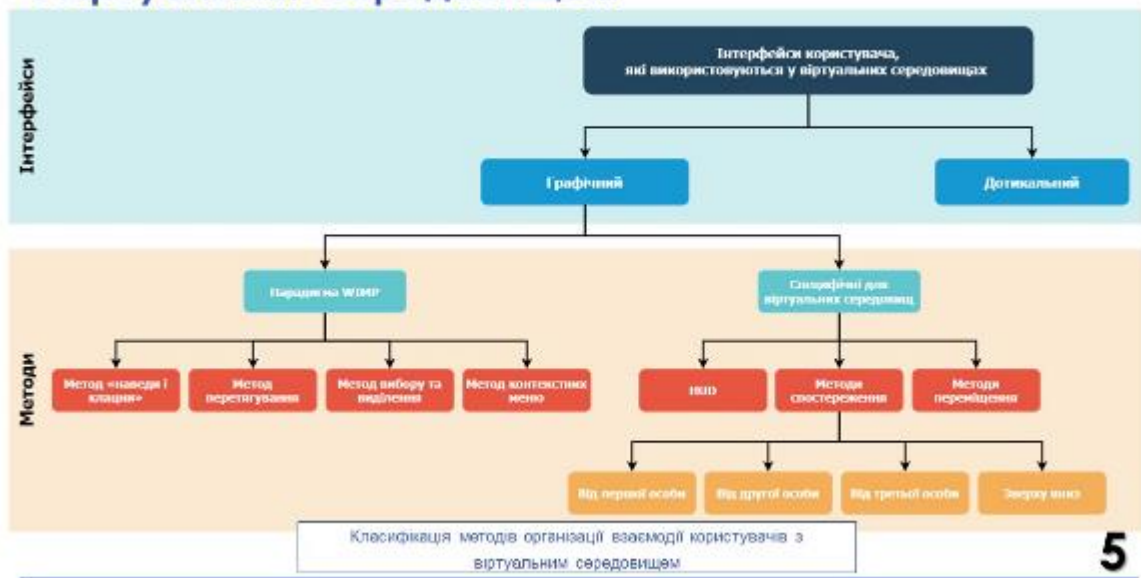


Рисунок Б.5 – Дослідження методів організації взаємодії користувачів з віртуальним середовищем

Удосконалений метод організації взаємодії користувачів з онлайн ВС Розумний будинок



Рисунок Б.6 – Удосконалений метод організації взаємодії користувачів з онлайн ВС Розумний будинок

Аналіз технологій реалізації ВС

Критерій \ Інструмент	MATLAB (Simulink)	Unity	Unreal Engine 4	WebGL, JS-фреймворки
Платформи, на яких працює	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows, Linux, MacOS	Усі веббрауери, які підтримують HTML5
Платформи, для яких можна збирати продукт	ОС, на яких працює сервер протоколу жва	ОС, на яких працює GE, мобільні ОС (Android, iOS), браузер, консоль, WebGL-процесор, веб-платформа	ОС, на яких працює GE, мобільні ОС (Android, iOS), браузер, консоль, WebGL-процесор, веб-платформа	—
Граничні можливості	Дуже багато 3D, менше 2D-можливості, які не забезпечують онлайнового рендерингу	Великий набір рендерингу 3D-графіки, світла та трафіку ефектів, що згоді забезпечують онлайновий рендеринг	Дуже багато 3D-графіки, наприклад, ефекти, які неможливо реалізувати в інших. Це досягається за допомогою оптимізації світла, розподілу роботи апаратного забезпечення на об'єкти ефектів	Дуже великі можливості рендерингу 3D-графіки, проте для його програмування може знадобитися багато часу
Мова програмування	MATLAB, Python, C/C++, Fortran, Java	C#, JavaScript, UnityScript, Bolt	C++ Blueprint	JavaScript, TypeScript
Тарифний план	Платі за ліцензією	Безкоштовно за ліцензією Personal та платі за версіями Pro, Plus та Enterprise	Безкоштовно, необхідно зареєструватися, платі за версіями Pro, Plus та Enterprise	Безкоштовно, але деякі фреймворки платі за версіями Pro, Plus та Enterprise
Граничні можливості	Великі можливості	Великі можливості	Великі можливості	Великі можливості
Особливості	Надає можливість розробити для MATLAB-розробників	Добре підходить для розробки сайтів та серверів	Великі можливості для розробки сайтів та серверів	Великі можливості для розробки сайтів та серверів
Підтримка експертних знань, інструментів, інструментів, інструментів	—	—	—	—
Підтримка експертних знань, інструментів, інструментів	+	+	+	+

7

Рисунок Б.7 – Аналіз технологій реалізації ВС

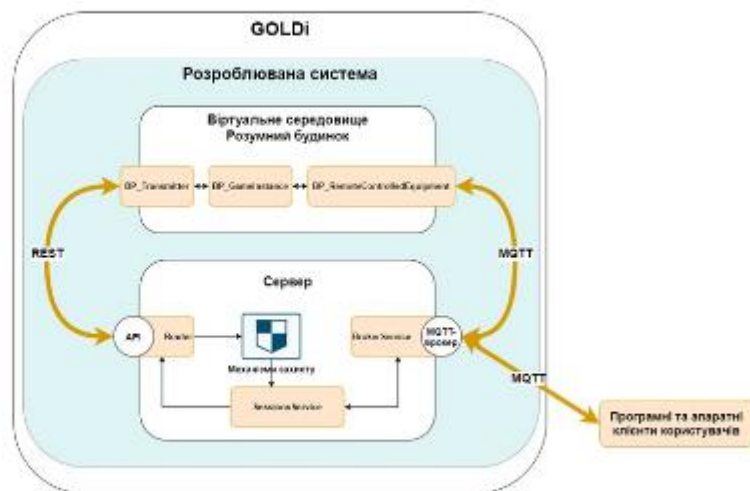
Аналіз протоколів передачі даних, що використовуються в системах Розумний будинок та IoT

Критерій \ Протокол	CoAP	MQTT	XMPP	AMQP
Типи зв'язку: синхронний/асинхронний	Синхронний та асинхронний	Асинхронний	Синхронний та асинхронний	Асинхронний
Продовжити реалізацію часу	+	+	+	—
Надійшли	Великий набір можливостей, які неможливо реалізувати в інших. Це досягається за допомогою оптимізації світла, розподілу роботи апаратного забезпечення на об'єкти ефектів	Великий набір можливостей, які неможливо реалізувати в інших. Це досягається за допомогою оптимізації світла, розподілу роботи апаратного забезпечення на об'єкти ефектів	Великий набір можливостей, які неможливо реалізувати в інших. Це досягається за допомогою оптимізації світла, розподілу роботи апаратного забезпечення на об'єкти ефектів	Великий набір можливостей, які неможливо реалізувати в інших. Це досягається за допомогою оптимізації світла, розподілу роботи апаратного забезпечення на об'єкти ефектів
Безпека	DTLS	SSL / TLS	SSL / TLS	SSL / TLS
Підтримка мережі: мобільна/фіксована (3G/4G)	+	+	—	+
Архітектура	Затит / одностороння	Публікація / підписка	Публікація / підписка, затит / одностороння	Публікація / підписка
Протокол транспортного рівня	UDP	TCP	TCP	TCP
Специфікація: стандарт	Зменшує споживання енергії	Зменшує споживання енергії	Зменшує споживання енергії	Зменшує споживання енергії
Продуктивність: швидкість, надійність	Зменшує споживання енергії	Зменшує споживання енергії	Зменшує споживання енергії	Зменшує споживання енергії

8

Рисунок Б.8 – Аналіз протоколів передачі даних, що використовуються в системах Розумний будинок та IoT

Архітектура програмної системи



9

Рисунок Б.9 – Архітектура програмної системи

Взаємодія компонентів програмної системи



10

Рисунок Б.10 – Взаємодія компонентів програмної системи

Архітектура API

Точки входу серверу					Структура MQTT-тем обладнання віртуального середовища	
Точка входу	Метод	Вхідні параметри	Відповідь	Призначення	Тема	Структура
session	GET	—	host: string; deviceId: number; sessionId: string; timeLeft: number; timeToDecide: number	Розпочати новий сеанс	Вітання, тема	дентифікатор_сеансу/hello
session/start_session/session_id	POST	sessionId	sessionId: string; isConfirmed: boolean; timeLeft: number	Підтвердити сеанс	SYN-тема	дентифікатор_сеансу/\$SYN...
session/finish_session/session_id	POST	sessionId	sessionId: string; isFinished: boolean	Закрити сеанс	Обладнання	дентифікатор_сеансу/identifikator_obladnaniya/ тип_обладнання/дентифікатор_обладнання...
session/decide_session_id	POST	sessionId	host: string; deviceId: number; sessionId: string; timeLeft: number; timeToDecide: number	Повторити/закінчити сеанс. БО може використовувати цей запит для розв'язання проблеми інтернет-з'єднання	Підписка на усі нові сеанси	дентифікатор_сеансу/M
session/start_session/finish_session_id	POST	sessionId	sessionId: string; isFinished: boolean	Запустити таймер сеансу. БО використовує цей запит для зняття таймера на сервері, після чого, якщо отримавши відповідь від MQTT-брокера	Підписка на усі теми, отримані у каналі	дентифікатор_сеансу/identifikator_obladnaniya/ тип_обладнання/M

11

Рисунок Б.11 – Архітектура API

Вибір інструментарію для розробки серверу

Порівняння технологій для backend-розробки					Порівняння середовищ розробки для Node.js-розробки				
Критерій / Технологія	Node.js / Express.js	Django	Laravel	Ruby on Rails	Критерій / IDE	IntelliJ IDEA	Visual Studio Code	Atom	Sublime Text 3
Тип	Фреймворк / Фреймворк	Фреймворк	Фреймворк	Фреймворк	Гарячий пуск	Підтримка	Безплатно	Безплатно	Безплатно
Мова програмування	JavaScript, TypeScript	Python	PHP	Ruby	Підтримка TypeScript	+	+	+	+
Рівень зручності роботи	2010	2005	2011	2004	Підтримка ОС	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows, Linux, MacOS
Архітектура	MVC	MVC	MVC	MVC	Вартість використання та підтримки	Платна	Платна	Платна	Платна
Вартість використання	Дуже простий	Простий	Простий	Простий	Інструмент запуску системи та підтримки	+	+	Потрібно встановити окремі плагіни	Потрібно встановити окремі плагіни
Орієнтований на розробку	Для малих та великих проєктів	Для розробки великих проєктів	Для малих та великих проєктів	Для малих та великих проєктів	Розширюваність	+	+	+	+
Швидкість розробки	Швидко	Швидко	Швидко	Швидко	Швидкість запуску	Платна	Швидко	Платна	Швидко
Продуктивність	Висока	Висока	Висока	Висока	Стабільність роботи	Стабільно працює	Стабільно працює	Платна	Стабільно працює
Вартість використання	Висока	Висока	Висока	Висока	Вартість розробки та підтримки	+	+	+	+
Тип ліцензії	MIT	MIT	MIT	MIT					

12

Рисунок Б.12 – Вибір інструментарію для розробки серверу

Розроблені алгоритми

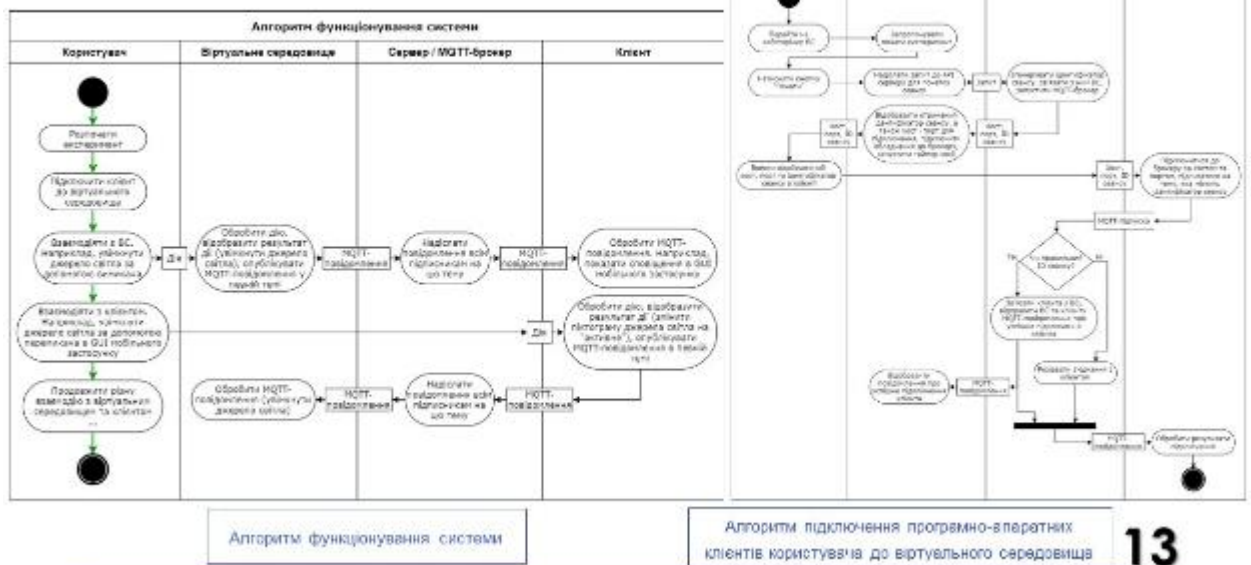


Рисунок Б.13 – Розроблені алгоритми

Інтерфейс ВС Розумний будинок



Рисунок Б.14 – Інтерфейс ВС Розумний будинок

Інтерфейс ВС Розумний будинок: Індикація стану



15

Рисунок Б.15 – Інтерфейс ВС Розумний будинок: Індикація стану

Висновки

- ✓ У роботі вирішено актуальну проблему розробки віртуального середовища Розумний будинок для гібридної лабораторії GOLDi для проведення комплексних експериментів з системами Розумний будинок та IoT.
- ✓ Наукова новизна отриманих результатів полягає у тому, що удосконалено метод організації взаємодії користувачів з віртуальним середовищем завдяки впровадженню методу віддаленого програмування для забезпечення нових способів отримання даних від користувача, підвищення зручності їх обробки, використання та простоти подання.
- ✓ Практична цінність одержаних результатів полягає у тому, що використання розробленого онлайн віртуального середовища Розумний будинок дозволяє викладачам створювати різні реалістичні завдання, пов'язані з функціонуванням систем Розумний будинок, а студентам – отримати досвід з проектування та розробки клієнтів для роботи з системами IoT та Розумний будинок, а також з розробки сценаріїв для домашньої автоматизації та оцінки їх ефективності.
- ✓ Основні положення й результати роботи доповідалися й обговорювалися на Міжнародній конференції ICL2021 24th International Conference on Interactive Collaborative Learning & 50th IGIP International Conference on Engineering Pedagogy, Дрезден, Німеччина, 22-24 вересня 2021 року. Також результати роботи були представлені в тезах доповіді НПК «Тиждень науки – 2021».

16

Рисунок Б.16 – Висновки