МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

КОМАР Вадим Андрійович

Моделі інтеграції голосових команд та жестів для управління "розумним" будинком / Models for integrating voice commands and gestures to control a smart home

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи Кім-21
В.А. Комар
_____

Науковий керівник
к.т.н., Г.М. Мельник
_____

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____О.Л. Дубчак

Тернопіль – 2024

АНОТАЦІЯ

Берегуля Ю.Р. Алгоритми аналізу та синтезу біомедичних зображень на основі фрактального підходу. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2024.

Робота написана обсягом 60 сторінок і містить 8 ілюстрацій, 3 таблиці, 2 додатки та 35 джерел за переліком посилань. Метою кваліфікаційної роботи є розроблення алгоритмів та моделі інтеграції голосових команд та жестів для управління "розумним" будинком.

Запропоновано комбіновані моделі інтеграції голосових команд і жестів, які дозволяють виконувати команди користувача через мультимодальне управління. Реалізовано алгоритми паралельної обробки голосу та жестів, що дозволяють системі синхронізувати результати різних модулів (розпізнавання тексту та ключових точок скелету). Розроблені моделі забезпечують точність і швидкість розпізнавання, мінімізацію помилок у визначенні команд, а також підтримку гнучких сценаріїв управління, таких як зміна параметрів за допомогою жестів або голосу.

Результати тестування алгоритму розпізнавання голосових команд показали ефективність системи у виконанні запитів, таких як отримання температури в конкретній кімнаті. Алгоритм успішно ідентифікував ключові параметри у висловлюванні користувача, обрав відповідний вузол системи , перевірив контекст на наявність усіх необхідних параметрів та сформував коректну команду для виконання. У випадках відсутності параметрів або помилок розпізнавання система забезпечувала користувача інформативними уточненнями або повідомленнями про неможливість виконання команди.

КЛЮЧОВІ СЛОВА: СИСТЕМИ РОЗУМНОГО БУДИНКУ, РОЗПІЗНАВАННЯ ГОЛОСУ, РОЗПІЗНАВАННЯ ЖЕСТІВ, MQTT.

ANNOTATION

Komar V.A. Models for integrating voice commands and gestures to control a smart home. - Manuscript.

Qualification work for obtaining the educational degree "Master" in specialty 123 "Computer Engineering", educational and professional program. West Ukrainian National University, Ternopil, 2024.

The work is 60 pages long and contains 8 illustrations,3 tables, 2 applications and 35 sources for references. The purpose of the qualification work is to develop algorithms and a model for integrating voice commands and gestures for controlling a "smart" home.

Combined models for integrating voice commands and gestures are proposed, which allow executing user commands through multimodal control. Parallel voice and gesture processing algorithms are implemented, which allow the system to synchronize the results of different modules (text recognition and key points of the skeleton). The developed models provide accuracy and speed of recognition, minimize errors in determining commands, and support flexible control scenarios, such as changing parameters using gestures or voice.

The results of testing the voice command recognition algorithm showed the effectiveness of the system in executing requests, such as obtaining the temperature in a specific room. The algorithm successfully identified key parameters in the user's utterance, selected the appropriate system node, checked the context for the presence of all necessary parameters, and formed the correct command for execution. In cases of missing parameters or recognition errors, the system provided the user with informative clarifications or messages about the impossibility of executing the command.

KEYWORDS: SMART HOME SYSTEMS, VOICE RECOGNITION, GESTURE RECOGNITION, MQTT.

ЗМІСТ

ВСТУП

Актуальність. Управління системами «розумного» будинку є одним із ключових напрямів розвитку сучасних технологій Інтернету речей (IoT). Зростаюча складність таких систем потребує впровадження інтуїтивно зрозумілих та ефективних способів взаємодії між користувачами та пристроями. Інтеграція голосових команд та жестів як методів управління відкриває нові можливості для підвищення зручності, доступності та універсальності таких систем.

Актуальність теми дослідження зумовлена швидким розвитком технологій розпізнавання голосу і жестів, які знаходять застосування у багатьох галузях, включаючи автоматизацію будинків. Взаємодія на основі голосу є природним і зручним для користувачів способом керування, що дозволяє забезпечити швидкий доступ до функцій системи. Жести, зі свого боку, є ефективним інструментом у випадках, коли голосове управління недоцільне або неможливе (наприклад, у шумному середовищі). Однак інтеграція цих методів у єдину систему вимагає розробки нових моделей і алгоритмів, здатних забезпечити узгодженість і надійність роботи.

Сьогодні існує широкий спектр рішень для окремого використання голосового або жестового управління, проте їх комплексна інтеграція в системах «розумного» будинку залишається недостатньо дослідженою. Виклики включають забезпечення синхронної обробки команд, мінімізацію затримок у передачі та виконанні даних, а також урахування контексту для коректного розпізнавання комбінованих запитів.

Реалізація системи інтеграції голосових команд та жестів для управління «розумним» будинком на локальному сервері стикається з низкою технічних викликів, серед яких однією з ключових проблем є обмеженість обчислювальних ресурсів. Сучасні алгоритми розпізнавання голосу та жестів, що базуються на методах глибокого навчання, вимагають значних обчислювальних потужностей для забезпечення високої точності та швидкості обробки даних.

Для розпізнавання голосових команд необхідно обробляти великі обсяги аудіоданих у реальному часі, що передбачає застосування складних моделей, таких

як рекурентні нейронні мережі (RNN) або трансформери. Аналогічно, розпізнавання жестів включає в себе обробку відеопотоку для виявлення ключових точок скелету, положення рук та пальців, що також є ресурсозатратним завданням. Локальні сервери, які зазвичай мають обмежену обчислювальну потужність та доступ до пам’яті, не завжди здатні підтримувати ефективну роботу цих алгоритмів у режимі реального часу.

Крім того, для одночасного розпізнавання голосу і жестів у комбінованих командах потрібна паралельна обробка кількох потоків даних. Це створює додаткове навантаження на локальний сервер, особливо якщо система обслуговує кількох користувачів одночасно. У таких умовах існує ризик затримок у виконанні команд, що може суттєво вплинути на зручність використання системи.

Обмеженість ресурсів також обмежує можливості використання сучасних методів забезпечення контекстуальної обробки, які потребують великих обсягів оперативної пам’яті для зберігання проміжних даних. У результаті система може втрачати контекст попередніх команд, що знижує її ефективність.

Метою дослідження є розроблення моделей та алгоритмів інтеграції голосових команд та жестів для управління системами «розумного» будинку, які дозволяють підвищити ефективність і зручність використання таких систем. У процесі дослідження проведено аналіз сучасних технологій керування голосом і жестами, визначено методи та засоби їх розпізнавання, а також запропоновано підходи до інтеграції в архітектуру «розумного» будинку.

Практична значущість роботи полягає у створенні програмного забезпечення для управління системою «розумного» будинку, що інтегрує голосові та жестові команди, а також у проведенні експериментальних досліджень для оцінки ефективності розроблених моделей і алгоритмів.

Метою кваліфікаційної роботи є розроблення алгоритмів та моделі інтеграції голосових команд та жестів для управління "розумним" будинком.

Об'єкт дослідження: процеси управління системами «розумного» будинку з використанням голосових команд і жестів..

Предмет дослідження: моделі, алгоритми та програмні засоби інтеграції голосових команд і жестів.

Задачі дослідження:

1. здійснити аналітичний огляд технологій керування з допомогою жестів та голосу;
2. розробити архітектуру системи управління "розумним" будинком;
3. розробити алгоритми та Моделі інтеграції голосових команд та жестів;
4. провести програмну реалізацію та експериментальне дослідження алгоритмів.

Наукова новизна одержаних результатів. Розроблено алгоритми та моделі інтеграції голосових команд та жестів із використанням контексту та шаблонів, що забезпечило ідентифікацію ключових параметрів у висловлюванні користувача.

Практичне значення отриманих результатів. Розроблені алгоритми дозволяють покращити користувацький досвід та розширити можливості автоматизації, забезпечити незалежність від зовнішніх мереж.

Апробація роботи. Отримані результати опубліковані в межах Всеукраїнської науково-практичної конференції «Інтелектуальні комп’ютерні системи та мережі», опубліковано тези доповіді [1,2].

Кваліфікаційна робота містить три розділи, висновки, список використаних джерел та додатки.

Перший розділ, присвячений аналізу методів управління системами «розумного» будинку.

Другий розділ присвячено розробці архітектури системи, розробленню детальної діаграми класів та розробленню алгоритмів інтеграції голосових команд і жестів.

Третій розділ присвячений розробці програмного забезпечення та проведенню експериментальних досліджень.

1. АНАЛІЗ МЕТОДІВ УПРАВЛІННЯ СИСТЕМАМИ «РОЗУМНОГО» БУДИНКУ

1.1 Аналіз технологій керування з допомогою жестів та голосу

Для керування складними системами з дискретними станами застосовується теорія автоматів і логічного управління. Вона використовується для розробки команд, які активують певні процеси залежно від стану системи.

У сучасних рішеннях для автоматизації та управління "розумним" будинком голосові команди та жести набувають дедалі більшої популярності як інструменти взаємодії. Ці технології створюють природний та інтуїтивний спосіб керування різноманітними пристроями і сервісами, забезпечуючи зручність та легкість використання.

Загальна архітектура систем розпізнавання голосу.

Системи розпізнавання голосу характеризуються чітко структурованою архітектурою, яка включає кілька ключових етапів обробки. Початковим компонентом є прийом вхідного звукового сигналу, який здійснюється за допомогою мікрофона або масиву мікрофонів. Після цього проводиться попередня обробка сигналу, що включає фільтрацію шуму та нормалізацію для підвищення якості даних. На наступному етапі здійснюється виділення ознак, де звуковий сигнал перетворюється на набір числових параметрів, таких як коефіцієнти Мел-кепстральних частот (MFCC) [2].

Акустична модель використовується для встановлення відповідності між виділеними ознаками та фонемами мови, тоді як лінгвістична модель враховує граматичні та статистичні особливості мови для формування ймовірних послідовностей слів. Завершальним етапом є декодування, яке визначає найбільш ймовірну послідовність слів на основі інтеграції акустичної та лінгвістичної моделей. Результат розпізнавання передається у вигляді команд до системи управління, наприклад, для автоматизації функцій "розумного" будинку.

Технології та платформи розпізнавання голосу.

Сучасні системи розпізнавання голосу базуються на різноманітних технологіях, які можна поділити на хмарні сервіси, локальні рішення та гібридні

підходи. Хмарні сервіси, такі як Amazon Alexa, Google Assistant та Microsoft Cortana, використовують потужності централізованих обчислювальних платформ для обробки голосових команд. Ці системи забезпечують високу точність завдяки масштабним моделям і великим обсягам тренувальних даних.

Локальні рішення, зокрема PocketSphinx та Kaldi, надають можливість виконання розпізнавання голосу без підключення до хмари. Ці бібліотеки дозволяють створювати автономні системи, що є важливим для зниження залежності від інтернет-з’єднання та забезпечення конфіденційності даних.

Гібридні підходи поєднують переваги локальних і хмарних технологій. Наприклад, первинне розпізнавання може виконуватися на локальному пристрої, а подальша обробка — у хмарі для підвищення точності та швидкодії. Такий підхід забезпечує оптимальний баланс між продуктивністю, затримками та точністю.

Інтеграція з системами "розумного" будинку.

Для забезпечення інтеграції систем розпізнавання голосу з платформами "розумного" будинку використовуються сучасні протоколи зв’язку, зокрема MQTT, HTTP та WebSockets. Ці протоколи дозволяють передавати команди до пристроїв, таких як освітлення, термостати або системи безпеки.

Автоматизація управління забезпечується за допомогою платформ, таких як Home Assistant [3], OpenHAB [4] або Node-RED, які слугують центральним вузлом для інтеграції різноманітних пристроїв і сервісів. Використання цих інструментів дозволяє створювати гнучкі та масштабовані системи для автоматизації функцій "розумного" будинку, забезпечуючи високу ефективність і зручність для користувачів.

Розглянемо детальніше протколи.

MQTT (Message Queuing Telemetry Transport) [5] – це легкий протокол обміну повідомленнями, призначений для передачі даних між пристроями в умовах обмежених ресурсів, нестабільного зв'язку чи низької пропускної здатності мережі. MQTT широко використовується в системах "розумного" будинку, Інтернету речей (IoT), телеметрії та інших застосунках, де потрібна надійна та ефективна передача даних. MQTT працює за моделлю публікація-підписка (Publish-Subscribe). Ця модель передбачає, що клієнти можуть або публікувати повідомлення, або підписуватися на

певні теми (топіки), але не зобов’язані знати одне про одного. Весь обмін даними відбувається через сервер, який називається брокером. Основні кроки роботи протоколу виглядають так:

1. Підключення - клієнт встановлює TCP-з'єднання з брокером MQTT і автентифікується за потреби.

2. Підписка (Subscribe) - клієнт може підписатися на певні теми (topic), які його цікавлять. Теми організовані ієрархічно, наприклад: home/kitchen/temperature.

3. Публікація (Publish) - інші клієнти публікують повідомлення на задані топіки, і брокер розсилає ці повідомлення всім клієнтам, які підписані на відповідні топіки.

4. Доставка повідомлень - брокер гарантує доставку повідомлень залежно від встановленого рівня якості обслуговування (QoS).

5. Розрив з’єднання - клієнт може закрити з'єднання, але брокер може зберігати останнє повідомлення для передачі новим підписникам, якщо це передбачено параметрами.

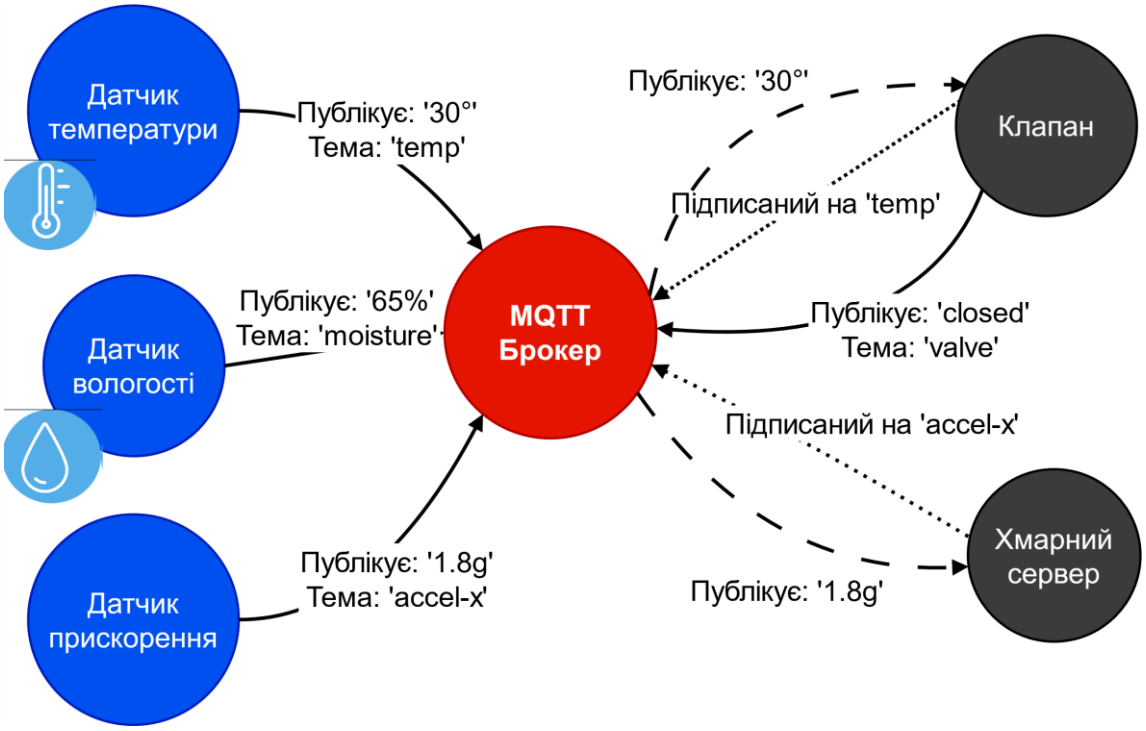


Рисунок 1.1 – Схема MQQT протоколу

Клієнти - це пристрої чи програми, які підключаються до брокера MQTT. Клієнти можуть бути публікаторами (Publisher) або підписниками (Subscriber).

Наприклад, термостат у "розумному" будинку може публікувати поточну температуру, а мобільний додаток підписується, щоб отримувати ці дані. Сервери - в MQTT сервер називається брокером. Брокер приймає повідомлення від клієнтів-публікаторів і розсилає їх клієнтам-підписникам, які зацікавлені у відповідних топіках. Теми (topic) – це текстові мітки, які визначають категорії або групи для повідомлень. Вони організовані в ієрархічну структуру, що полегшує фільтрацію та маршрутизацію даних.

1.2 Методи і засоби розпізнавання жестів

Технології розпізнавання жестів значно просунулися, особливо завдяки інтеграції інерційних датчиків, таких як акселерометри та гіроскопи. Ці датчики є ключовими для збору даних про рух, що дозволяє здійснювати розпізнавання жестів у реальному часі для різних застосувань, включаючи взаємодію людини з машинами.

У цьому дослідженні [6] представлено метод безперервного розпізнавання жестів рук за допомогою трьохосьового акселерометра та гіроскопа. Підкреслюється розробка прототипу системи, здатної до реального розпізнавання жестів, що сприяє ефективній взаємодії людини з машинами.

У цій статті [7] обговорюються алгоритми, які класифікують послідовності жестів рук на основі даних з акселерометрів і гіроскопів. Дослідження зосереджено на підвищенні точності систем розпізнавання жестів, які можуть бути застосовані в різних споживчих електронних пристроях.

У цій статті [8] досліджується носимий пристрій, який використовує як акселерометри, так і гіроскопи для розпізнавання лінійних і обертальних жестів. Підкреслюється важливість цих датчиків у покращенні точності та чутливості систем розпізнавання жестів.

Дослідження [9] оцінює методи машинного навчання в поєднанні з датчиками руху, включаючи акселерометри, для розпізнавання рухів зап'ястя. Результати сприяють розумінню того, як носима технологія може покращити взаємодію на основі жестів.

Дослідження [10] вивчає функціональність акселерометра та гіроскопа в мобільних застосунках для взаємодії на основі жестів, підкреслюючи їх потенціал у покращенні користувацького досвіду через інтуїтивне управління.

Система домашньої автоматизації [11] забезпечує доступ до побутової техніки та є корисною для людей з різними проблемами та людей похилого віку, які мають проблеми з пересуванням. Доведено, що система домашньої автоматизації робить людей з обмеженими можливостями та людей похилого віку незалежними, а також робить їхнє життя легшим і стандартизованішим. Система Home Automation на основі рукавичок реалізована за допомогою методу розпізнавання жестів. Пацієнти керують приладами, обробляючи прості жести руками, які виконуються в рукавичках. Сторож керує технікою через Google Assistant і мобільний додаток. Разом з цим вони можуть переглядати поточний стан пристроїв, а також час і дату попередньої дії пацієнта. Таким чином, система є ефективною, економічною та енергозберігаючою.

Рисунок 1.2 – Пристрої «розумного» будинку

Рисунок 1.2 представляє концепцію «розумного» будинку (Smart Home), яке ілюструє інтеграцію різноманітних пристроїв для автоматизації побутових завдань і підвищення комфорту та безпеки житлового приміщення.

На зображенні показані такі компоненти, які можуть бути інтегровані в систему розумного дому:

1. Освітлення — лампи, які можуть бути віддалено керовані через мобільний додаток або автоматично налаштовуватися відповідно до присутності мешканців.
2. Електроприлади — холодильники, телевізори, та інша побутова техніка, яка може бути під'єднана до розумної системи для зручного керування.
3. Система відеоспостереження — камери безпеки для моніторингу території та підвищення безпеки.
4. Опалення та кондиціювання — контролери температури, які забезпечують автоматичне налаштування температури для комфортного проживання.
5. Датчики (температури, руху, вологості) — для моніторингу навколишнього середовища та прийняття відповідних дій (наприклад, увімкнення опалення чи кондиціонера).
6. Вентиляція та контроль якості повітря — вентиляційні пристрої для регулювання якості повітря.
7. Керування замками — електронні замки для підвищення безпеки, якими можна керувати дистанційно.
8. Пральна машина та інші побутові пристрої — автоматизація процесів прання та інших побутових завдань.

На ілюстрації також зображено смартфон, що показує, як можна керувати всіма цими пристроями через єдиний інтерфейс або мобільний додаток.

У контексті розробки ця схема показує можливості взаємодії між різними пристроями через IoT (Інтернет речей) інтерфейси та мережеві технології. Для розробників такі системи включають такі важливі аспекти, як:

- програмна платформа для контролю та автоматизації (наприклад, через хмарні сервіси);
- безпека передачі даних між різними компонентами, особливо враховуючи, що багато з них мають критично важливі функції;

– інтерфейси та протоколи зв'язку, як-от Zigbee, WiFi або BLE, які забезпечують взаємодію між датчиками, актуаторами та центральним контролером (наприклад, смартфоном або сервером).

Рисунок 1.3 представляє архітектуру системи автоматизації для домашніх приладів із використанням технології ZigBee. Схема показує, як користувач взаємодіє з розумними домашніми пристроями через смартфон і сенсорний вимикач.

Рисунок 1.3 – Схема системи автоматизації і ємнісними вимикачами

Опис компонентів системи:

1. Домашні прилади (Home Appliances) — електронні прилади, які можуть вмикатися та вимикатися за допомогою системи керування.

2. Реле (Relay) — електронний компонент, який здійснює включення/вимкнення приладів. Воно отримує сигнали від контролера і виконує операцію "ON/OFF" для домашнього приладу.

3. Ємнісний сенсорний вимикач (Capacitive Touch Switch) — вимикач, яким користувач може фізично керувати, щоб вмикати або вимикати пристрої. Він надсилає сигнал контролю до реле.

4. Модуль ZigBee — пристрої, які забезпечують бездротовий зв'язок між елементами системи. ZigBee модулі взаємодіють із реле, сенсорним вимикачем та іншими компонентами для передачі даних.

5. ZigBee мережа (ZigBee Network) — мережа, яка використовується для передачі команд та даних між різними компонентами системи, включаючи модуля ZigBee, реле та контролери.

6. Користувач (User) — людина, яка взаємодіє із системою через сенсорний вимикач або смартфон.

7. Смартфон (Smart Phone) — використовується як засіб віддаленого керування пристроями через спеціальний додаток. Смартфон взаємодіє з мережею ZigBee для отримання інформації та відправки команд.

8. MCU (Мікроконтролерний блок управління) — центральний мікроконтролер, який відповідає за координацію і контроль роботи системи, приймаючи команди від користувача та виконуючи їх.

9. Платформа управління (Smart Management Platform) — програмна платформа, яка дозволяє користувачеві контролювати та моніторити роботу приладів за допомогою смартфона через мережу ZigBee та MCU.

Опис процесу роботи:

- користувач може взаємодіяти із системою як за допомогою сенсорного вимикача, так і через смартфон;
- ємнісний сенсорний вимикач може вмикати або вимикати прилади через реле, яке отримує команду;
- модулі ZigBee з'єднують різні компоненти в ZigBee мережі для бездротового зв'язку, забезпечуючи комунікацію між контролером, реле та іншими пристроями;
- смартфон дозволяє користувачу контролювати домашні прилади віддалено через модуль ZigBee і MCU;
- MCU забезпечує обробку даних і комунікацію з платформою управління, яка слугує інтерфейсом для налаштування та моніторингу системи.

Переваги інтерфейсу системи:

1. Віддалене керування через смартфон. Користувачі можуть керувати домашніми приладами з будь-якого місця, використовуючи смартфон. Можна оперативно відслідковувати стан пристроїв і приймати необхідні рішення (включення/вимкнення) у реальному часі.

2. Фізичний контроль через сенсорний вимикач. Фізичний сенсорний вимикач дозволяє вмикати та вимикати пристрої без використання смартфона, що зручно для всіх користувачів, включаючи тих, хто не звик користуватися мобільними додатками. Сенсорний вимикач забезпечує інтуїтивне управління, яке не потребує спеціальних знань.

3. Надійний зв'язок через ZigBee мережу. Використання технології ZigBee дозволяє знизити енергоспоживання, що робить систему ефективною. ZigBee забезпечує стабільний зв'язок у локальній мережі з низьким рівнем затримок, що підходить для застосування в системах автоматизації.

4. Зворотний зв'язок для моніторингу стану пристроїв. Система забезпечує зворотний зв'язок від реле до користувача, що дозволяє відстежувати стан пристроїв і впевнюватися, що команда виконана.

Недоліки інтерфейсу системи:

1. Залежність від смартфона та інтернет-з'єднання. Якщо смартфон не має підключення до інтернету, можливість керування пристроями з дистанції стає недоступною. Якщо є проблеми з програмою на смартфоні (наприклад, помилки в додатку або збої), це може обмежити можливість керування.

2. Складність для менш досвідчених користувачів. Для початкового налаштування системи необхідні певні знання в роботі з мобільними додатками та технологіями IoT, що може стати складністю для користувачів, які не мають технічного досвіду. Для повноцінного користування потрібно мати базові знання про те, як працює ZigBee мережа, що також може бути викликом для користувачів без технічного досвіду.

3. Залежність від наявності електроенергії. Електронні компоненти, такі як реле та модулі ZigBee, залежать від стабільного електропостачання, що може бути проблемою при перебоях у живленні або аваріях.

4. Потенційні проблеми безпеки. Використання бездротового зв'язку і мобільного додатку створює потенційну загрозу для безпеки системи, особливо якщо не використовуються належні заходи захисту даних. ZigBee є досить безпечною технологією, проте при недостатніх заходах захисту можливі атаки, наприклад, перехоплення команд або несанкціонований доступ до пристроїв.

1.3 Інтеграція з системами "розумного" будинку

Розглянемо протоколи зв'язку MQTT, HTTP, WebSockets для передачі команд до пристроїв.

MQTT — це легкий протокол обміну повідомленнями, який був розроблений для пристроїв з обмеженими ресурсами та для нестабільних мереж. Він часто використовується в "розумних" будинках через здатність ефективно передавати дані в умовах низької пропускної здатності, що робить його ідеальним для пристроїв IoT. Функції MQTT:

- Відправка та отримання даних забезпечує обмін повідомленнями між клієнтами через брокера.
- Реалізація моделі видавець/підписник (Pub/Sub) - пристрої можуть підписуватися на певні теми або публікувати дані на них.
- Низька затримка - підтримує швидкий обмін даними з мінімальною затримкою.
- Надійність передачі - Забезпечення три рівня якості обслуговування (QoS) для гарантії доставки повідомлень.

Архітектура:

- Брокер MQTT - це центральний сервер, який координує обмін повідомленнями між видавцями та підписниками. Він забезпечує маршрутизацію повідомлень, що надходять від видавців, до відповідних підписників.
- Клієнти – це Пристрої або системи, що можуть бути як видавцями, так і підписниками. Наприклад, датчик температури може бути видавцем, а мобільний додаток — підписником на тему "температура".

Архітектурні особливості:

- Взаємодія через Теми (Topics) - дані публікуються на певну тему, наприклад, "розумний_будинок/освітлення/вітальня". Пристрої, які підписуються на цю тему, отримують повідомлення.
- Модель асинхронної передачі - клієнти не взаємодіють напряму, що знижує вимоги до ресурсів та спрощує масштабування системи.

MQTT часто використовується для збирання даних з датчиків та для управління пристроями, такими як світло, термостати, замки тощо. Його легкість та низька затримка роблять його підходящим для ситуацій, коли пристрої мають обмежені ресурси або доступ до нестабільного з'єднання.

HTTP — основний протокол для передачі даних у веб-просторі. Він підходить для ситуацій, коли необхідно мати просте з'єднання для отримання або передачі даних між пристроями розумного будинку і сервером або користувачем.

Функції HTTP:

– Запит-відповідь - HTTP використовує модель запит-відповідь, де клієнт відправляє запит серверу, а сервер відповідає даними.

– REST API - HTTP часто використовується разом з RESTful API для інтеграції компонентів "розумного" будинку.

– Простота використання - HTTP легко інтегрується з іншими веб-технологіями, що робить його ідеальним для інтерфейсів користувача.

Архітектура HTTP:

– клієнт-серверна архітектура - HTTP працює за моделлю клієнт-сервер, де клієнт надсилає запит до сервера, а сервер відповідає даними;

– стани - HTTP є безстанним протоколом, що означає, що кожен запит виконується незалежно від попередніх.

Архітектурні особливості:

– HTTP-запити - GET, POST, PUT, DELETE. Наприклад, клієнт може надіслати запит GET для отримання даних з датчика або POST для зміни параметрів;

– Використання RESTful сервісів - сервери, які взаємодіють з пристроями "розумного" будинку, можуть використовувати RESTful підхід для виконання операцій.

HTTP використовується для надання доступу до веб-інтерфейсів керування "розумним" будинком, наприклад, для перегляду інформації про стан пристроїв або для ручного управління через мобільний додаток чи браузер. Він підходить для менше критичних до затримок процесів, таких як зміна налаштувань чи отримання даних про стан системи.

WebSockets [12] — це протокол, який забезпечує постійне двостороннє з'єднання між клієнтом і сервером, що дозволяє реальний час передачі даних без постійних запитів. Цей протокол є ефективним рішенням для інтерактивних систем "розумного" будинку, де необхідна миттєва реакція на події.

Функції протоколу WebSockets**:**

– двосторонній зв'язок - Забезпечує постійне з'єднання між клієнтом і сервером, дозволяючи як клієнту, так і серверу відправляти повідомлення в будь-який момент;

– Зниження навантаження - на відміну від HTTP, де кожен запит вимагає окремого з'єднання, WebSockets дозволяють підтримувати одне з'єднання для тривалої передачі даних;

– реальний час - Підходить для застосунків, що потребують миттєвого оновлення, наприклад, відеоспостереження чи інтерфейсів управління.

Архітектура WebSockets:

– з'єднання через HTTP - З'єднання WebSocket спочатку ініціюється через HTTP-запит, але потім переходить у постійне двостороннє з'єднання;

– постійне з'єднання - WebSockets забезпечують низьку затримку передачі даних завдяки постійному каналу зв'язку, який дозволяє передачу даних між клієнтом і сервером без переривань.

Архітектурні особливості WebSockets:

– фрейми даних - повідомлення в WebSockets передаються у вигляді фреймів, що можуть бути як текстовими, так і бінарними;

– обробка подій - сервер може відправляти дані в будь-який момент часу, що особливо корисно для динамічних сценаріїв.

WebSockets використовуються для забезпечення реального часу обміну даними між пристроями або користувачем та системою "розумного" будинку. Наприклад, вони підходять для сповіщень про зміну стану датчика або управління світлом через мобільний додаток без затримок. Завдяки своїй низькій затримці, WebSockets ідеально підходять для випадків, коли потрібно миттєво реагувати на зміну стану.

Таблиця 1.1 - Порівняння MQTT, HTTP та WebSockets

Критерій	MQTT	HTTP	WebSockets
Архітектура	Брокер та клієнти	Клієнт-сервер	Постійне двостороннє з'єднання
Модель передачі даних	Видавець/підписник (Pub/Sub)	Запит-відповідь	Двосторонній зв'язок
Надійність	QoS (1-3 рівня надійності)	Без гарантій доставки	Постійне з'єднання, підтвердження
Затримка	Низька	Середня (вимагає окремих запитів)	Низька
Використання	Сенсори, команди	Веб-інтерфейси, налаштування	Реальний час, миттєві події
Придатність	Обмежені ресурси	Прості запити, людські взаємодії	Інтерактивні програми, реальний час

1.4 Аналіз завдань магістерської роботи та постановка задач дослідження

Як видно з аналітичного огляду все більше уваги приділяється інтуїтивно зрозумілим способам взаємодії, які включають голосові команди та жести. Розпізнавання таких форм взаємодії дозволяє підвищити зручність і ефективність управління різноманітними пристроями та системами автоматизації. Актуальність цієї теми зумовлена потребою в локальних рішеннях, які забезпечують високу швидкість обробки команд, незалежність від зовнішніх мереж і високий рівень конфіденційності даних.

Метою роботи є розробка моделі інтеграції голосових команд і жестів для управління пристроями "розумного" будинку з використанням локального сервера. Модель повинна забезпечувати високу точність розпізнавання, мінімальну затримку

виконання команд і можливість масштабування для підтримки великої кількості пристроїв.

Для досягнення мети необхідно вирішити наступні завдання:

1. Розробити архітектуру локального сервера, здатного обробляти голосові команди та жести в реальному часі. Сервер повинен включати модулі збору, попередньої обробки даних і виділення ознак із вхідних сигналів голосу та відео.

2. Побудувати моделі для розпізнавання голосу та жестів з використанням методів машинного навчання та нейронних мереж. Для голосових команд планується використовувати алгоритми для обробки аудіосигналу, такі як спектральний аналіз та глибокі нейронні мережі. Для розпізнавання жестів передбачається застосування моделей комп'ютерного зору, включаючи аналіз ключових точок скелетона.

3. Інтегрувати моделі розпізнавання голосу та жестів у єдину систему управління, яка здатна обробляти різні типи команд та транслювати їх для виконання множиною пристроїв "розумного" будинку.

4. Розробити механізм зв’язку між сервером і пристроями через стандартні протоколи, такі як MQTT, HTTP чи WebSockets, для забезпечення надійного і швидкого виконання команд.

5. Реалізувати систему, яка дозволяє одночасно працювати з кількома пристроями "розумного" будинку, виконуючи інтегровані команди, що враховують як голосові інструкції, так і жести.

6. Провести тестування системи на реальних даних, оцінюючи точність розпізнавання голосу та жестів, затримку виконання команд, а також адаптивність і масштабованість розробленої моделі.

Результатом роботи має стати функціональна система, яка реалізує локальну обробку голосових команд і жестів для інтеграції та управління пристроями "розумного" будинку. Розроблена модель забезпечить підвищення ефективності систем автоматизації, їхню безпеку, конфіденційність даних та незалежність від зовнішніх хмарних сервісів.

1.5 Висновки до розділу 1

Кожен із протоколів — MQTT, HTTP і WebSockets — має свої унікальні особливості та оптимальні сфери застосування в контексті "розумного" будинку. MQTT вирізняється ефективністю для пристроїв із обмеженими обчислювальними ресурсами, забезпечуючи передачу повідомлень із мінімальною затримкою, що робить його ідеальним для задач реального часу. HTTP, у свою чергу, є оптимальним вибором для статичних операцій або процесів, які не потребують миттєвого реагування, наприклад, для доступу до даних через веб-інтерфейс. WebSockets забезпечують двосторонній обмін даними в режимі реального часу, що є ключовим для інтерактивних сценаріїв та швидкого реагування на події, характерних для сучасних автоматизованих систем.

2 РОЗРОБКА МОДЕЛЕЙ І АЛГОРИТМІВ ІНТЕГРАЦІЇ

2.1 Архітектура системи

Архітектура системи управління "розумним" будинком, яка використовує голосові команди та жести, повинна бути спроєктована з акцентом на забезпечення надійності, ефективності та можливості масштабування. При цьому важливо враховувати обмежені обчислювальні ресурси, характерні для таких пристроїв, як мікроконтролери, наприклад, Raspberry Pi.

Архітектура з чотирьох основних компонентів:

Рисунок 2.1 – Архітектура системи

Кожен з цих компонентів виконує певні функції та взаємодіє з іншими модулями для забезпечення мультимодального управління будинком.

Модуль 1 розпізнавання голосу виконує функцію прийому та обробки голосових команд користувача. Він отримує аудіосигнали, проводить їх обробку та перетворює на текстові команди, які передаються до модуля інтеграції для подальшої інтерпретації та виконання. Для реалізації цього компонента доцільно використовувати платформи, такі як PocketSphinx або Google Speech API. Вибір на користь PocketSphinx є оптимальним, оскільки ця система працює локально, не потребуючи постійного доступу до Інтернету. Це знижує залежність від мережевої інфраструктури та підвищує рівень безпеки.

Модуль 2 розпізнавання жестів призначений для аналізу рухів користувача та інтерпретації певних жестів для управління системою. У цьому процесі використовуються камери або інші сенсори, які фіксують рухи. Обробка отриманих зображень здійснюється за допомогою таких бібліотек, як OpenCV та MediaPipe. OpenCV надає базові інструменти для обробки візуальної інформації, тоді як MediaPipe дозволяє інтегрувати сучасні алгоритми для точного та ефективного розпізнавання жестів.

Модуль 3 інтеграції з пристроями забезпечує взаємодію з пристроями "розумного" дому за допомогою підтримуваних протоколів зв'язку, зокрема HTTP, MQTT або WebSockets. Серед цих протоколів, MQTT виділяється як оптимальний вибір для таких систем завдяки своїй легкості, надійності та здатності ефективно працювати навіть за умов нестабільного інтернет-з’єднання. Використовуючи модель публікації та підписки (Pub/Sub), MQTT спрощує масштабування системи і підключення нових пристроїв. Цей протокол особливо зручний для інтеграції з мікроконтролерами, такими як Raspberry Pi, що робить його незамінним компонентом для розгортання "розумних" домашніх систем.

Користувацький інтерфейс 4 надає можливість користувачу налаштовувати систему, керувати нею, а також отримувати інформацію про стан пристроїв. Для реалізації інтерфейсу можна використовувати Flask або Node.js. Flask є зручним фреймворком для створення веб-додатків, що дозволяє швидко і легко розробити інтерфейс для налаштування та моніторингу "розумного" будинку. Flask також є оптимальним для студентських проектів, оскільки має просту архітектуру та легку інтеграцію з іншими компонентами системи.

Протокол MQTT обраний через його легкість і можливість обробки повідомлень у режимі реального часу при обмеженій пропускній здатності мережі. Це важливо для проектів, де можуть бути обмежені ресурси, і необхідно забезпечити стабільну роботу навіть при нестабільному підключенні до інтернету.

OpenCV та MediaPipe обрані для реалізації модуля розпізнавання жестів, оскільки вони забезпечують високу точність розпізнавання при відносно низьких вимогах до апаратних ресурсів. Ці бібліотеки є популярними та добре документованими.

Flask обраний для реалізації користувацького інтерфейсу через його простоту та можливість швидкої інтеграції з іншими компонентами системи. Він дозволяє створювати інтуїтивно зрозумілі інтерфейси без значних витрат часу і ресурсів.

2.2 Розроблення діаграми класів

Створення діаграми класів для модуля обробки голосових команд, включаючи локальний варіант (PocketSphinx) та віддалений варіант (Google Speech API), було обґрунтовано необхідністю продемонструвати архітектурні особливості кожного з підходів. Кожен варіант має свої переваги та обмеження, які були враховані під час розробки діаграми, щоб чітко визначити структуру модулів, їх функції та взаємодію між компонентами системи.

Діаграма класів ілюструє об'єктно-орієнтований підхід до проєктування модуля обробки голосових команд, що дозволяє ефективно організувати код і спрощує його підтримку. Кожен компонент модуля має чітко визначені функції, що сприяє реалізації принципу розподілу обов’язків, підвищує читабельність коду та забезпечує гнучкість під час внесення змін. У розробленій системі передбачено два основних сценарії обробки голосових команд: локальний і віддалений, які відрізняються підходами до роботи з аудіоданими.

Локальний сценарій, реалізований за допомогою PocketSphinx, представлений класом LocalVoiceCommandModule. Цей клас є центральним елементом модуля, що взаємодіє з іншими класами, такими як MicrophoneInput, CommandProcessor і LocalRecognizer. LocalVoiceCommandModule відповідає за управління всіма процесами, пов’язаними з обробкою голосу, та забезпечує виконання команд на основі розпізнаних голосових введень. Головна перевага локальної системи розпізнавання, такої як PocketSphinx, полягає у зменшенні залежності від інтернет-з’єднання та забезпеченні високого рівня безпеки оброблюваних даних. Компоненти MicrophoneInput, CommandProcessor та LocalRecognizer виконують специфічні завдання. Клас MicrophoneInput виконує функцію отримання аудіосигналів,

забезпечуючи їх передачу для подальшої обробки. CommandProcessor займається аналізом цих даних, використовуючи локальний механізм розпізнавання (LocalRecognizer) для ідентифікації відповідних команд, що потребують виконання. Архітектурне рішення щодо використання окремих класів для кожного етапу обробки дозволяє знизити складність кожного компонента, що, у свою чергу, полегшує підтримку та тестування.

Віддалений підхід, заснований на використанні Google Speech API, реалізовано через клас RemoteVoiceCommandModule, який взаємодіє з MicrophoneInput та інтегрує клас CloudCommandProcessor. Цей компонент відповідає за обробку аудіосигналів за допомогою хмарного сервісу Google Speech API. Хоча RemoteVoiceCommandModule за структурою схожий на локальний модуль, його ключовою відмінністю є перенесення основних процесів обробки аудіоданих на серверну сторону. Клас CloudCommandProcessor виконує функції передачі аудіоданих до хмарного сервісу та отримання результатів розпізнавання для подальшої інтерпретації. Цей підхід забезпечує високу точність розпізнавання завдяки використанню потужних алгоритмів Google, що можуть бути важкими для реалізації на локальних пристроях з обмеженими ресурсами. Основне обґрунтування цього підходу полягає у зменшенні вимог до локальних ресурсів та використанні переваг сучасних хмарних технологій для розпізнавання мови. Проте цей підхід має певні обмеження, такі як залежність від наявності стабільного інтернет-з'єднання та необхідність передавання даних на сторонні сервери.

Використання окремих класів для локального та віддаленого варіантів розпізнавання дозволяє легко порівнювати обидва підходи та гнучко змінювати варіанти реалізації залежно від умов експлуатації. Такий дизайн дозволяє, наприклад, легко замінити локальну систему на хмарну або навіть поєднувати обидва підходи для досягнення більшої надійності. Це також відповідає принципу відкритості-закритості (Open/Closed Principle), згідно з яким модулі системи повинні бути відкритими для розширення, але закритими для модифікації, що забезпечує масштабованість та зручність впровадження нових функцій без необхідності значних змін у вже реалізованих компонентах.

Діаграма класів (рисунок 2/2) створює чітку структуру взаємодії між компонентами системи обробки голосових команд, забезпечуючи логічний поділ функціоналу та зрозумілість для майбутньої модернізації чи масштабування. Запропонована архітектура ефективно поєднує використання локальних і хмарних обчислювальних ресурсів, що гарантує надійне розпізнавання голосу навіть за умов обмеженої обчислювальної потужності. Це рішення ідеально відповідає потребам "розумного" будинку, з урахуванням його специфічних вимог до продуктивності та адаптивності системи.

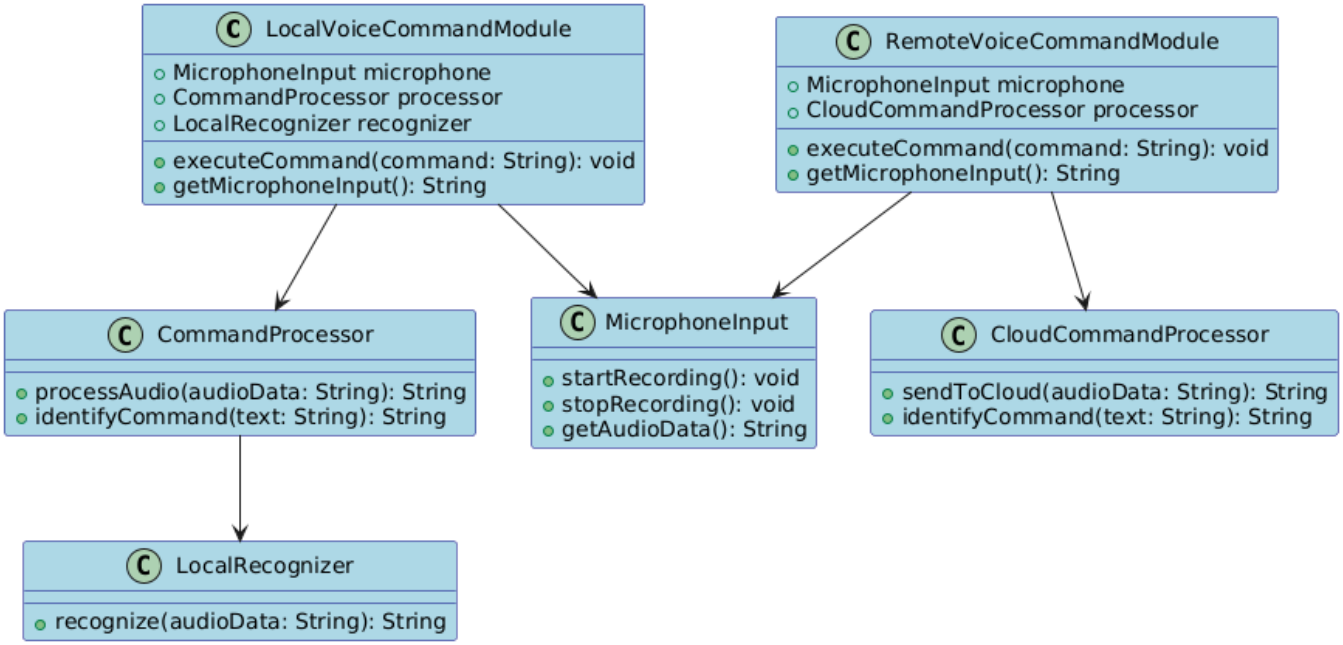


Рисунок 2.2 – Діаграма класів

Діаграма класів описує архітектуру системи обробки голосових команд, яка забезпечує модульність, чіткий розподіл функцій між компонентами та інтеграцію локального й віддаленого підходів для обробки аудіоданих.

Локальний модуль обробки голосових команд, представлений класом LocalVoiceCommandModule, взаємодіє з трьома основними компонентами: MicrophoneInput, CommandProcessor та LocalRecognizer. Клас MicrophoneInput відповідає за прийом аудіоданих, забезпечуючи функції початку та зупинки запису, а також отримання звукових сигналів. CommandProcessor займається обробкою цих даних, виконуючи аналіз аудіосигналу та визначення текстових команд на основі розпізнаних даних. Компонент LocalRecognizer забезпечує локальне розпізнавання

аудіосигналів, що дозволяє системі працювати без підключення до інтернету, підвищуючи безпеку та автономність. Основний модуль об'єднує ці компоненти для виконання команд користувача на базі отриманих голосових вказівок.

Віддалений модуль, представлений класом RemoteVoiceCommandModule, має схожу архітектуру, але замість локального розпізнавання використовує хмарний сервіс для аналізу аудіоданих. Він також взаємодіє з класом MicrophoneInput, а обробка аудіосигналів виконується через CloudCommandProcessor, який відповідає за передачу звукових даних до хмарного сервісу (наприклад, Google Speech API) і отримання результатів розпізнавання. Такий підхід забезпечує високу точність обробки голосу завдяки використанню потужних обчислювальних ресурсів сервера.

Архітектурна схема ілюструє переваги модульного підходу, забезпечуючи гнучкість та можливість адаптації до різних сценаріїв використання. Вона дозволяє реалізувати локальні та хмарні методи обробки, що є ключовим аспектом для забезпечення надійної роботи системи в умовах "розумного" будинку. Діаграма демонструє чіткі зв'язки між класами, спрощуючи реалізацію та підтримку системи, а також її розширення новими функціональними можливостями.

Додаймо нові класи та функції що реалізують функції по створенню і передачі команд на основі голосу і жестів.

Для реалізації нового класу, який відповідає за взаємодію із смарт-пристроями через API (наприклад, Tuya API), формує базу ймовірних команд і інтегрується з наявними класами, пропоную створити клас SmartDeviceCommandManager. Цей клас забезпечуватиме отримання доступних функцій пристроїв, їх обробку та передачу в модулі розпізнавання голосу. Нижче наведено реалізацію у вигляді діаграми класів UML:

Клас SmartDeviceCommandManager відповідає за автоматичне отримання доступних функцій і команд зі смарт-пристроїв. Він забезпечує зв’язок із Tuya API (або іншим API) для отримання актуальної інформації про пристрої та формує базу команд. Основні методи:

- fetchCommands() викликає функцію для підключення до API та отримання списку доступних команд.

– getAvailableCommands() повертає список ймовірних команд для інших модулів.

– connectToTuyaAPI() встановлює з’єднання із сервером Tuya API для отримання даних.

– processDeviceCommands(apiResponse: String) обробляє відповідь від API, аналізує та структурує її у вигляді списку команд.

Взаємодія із існуючими класами

– SmartDeviceCommandManager передає доступні команди в клас CommandProcessor, що дозволяє обробляти ці команди у системі розпізнавання.

– Інтеграція з LocalVoiceCommandModule забезпечує актуалізацію команд, доступних для виконання, покращуючи функціональність голосового управління.

Ця архітектура дозволяє адаптувати систему до нових пристроїв та розширювати її функціональність, зберігаючи модульність і гнучкість у реалізації. Для підтримки керування пристроями через жести, які розпізнаються на основі відео з камер, потрібно розширити функціональність класу SmartDeviceCommandManager. Додамо функції для обробки жестів і відправлення відповідних команд до смарт-пристроїв (рисунок 2.3).

Нові функції класу SmartDeviceCommandManager

– executeCommandByGesture(gesture: String) виконує команди для керування смарт-пристроями на основі розпізнаного жесту. Викликає mapGestureToCommand для визначення відповідної команди.

– mapGestureToCommand(gesture: String) перетворює розпізнаний жест на команду, яку розуміє смарт-пристрій.

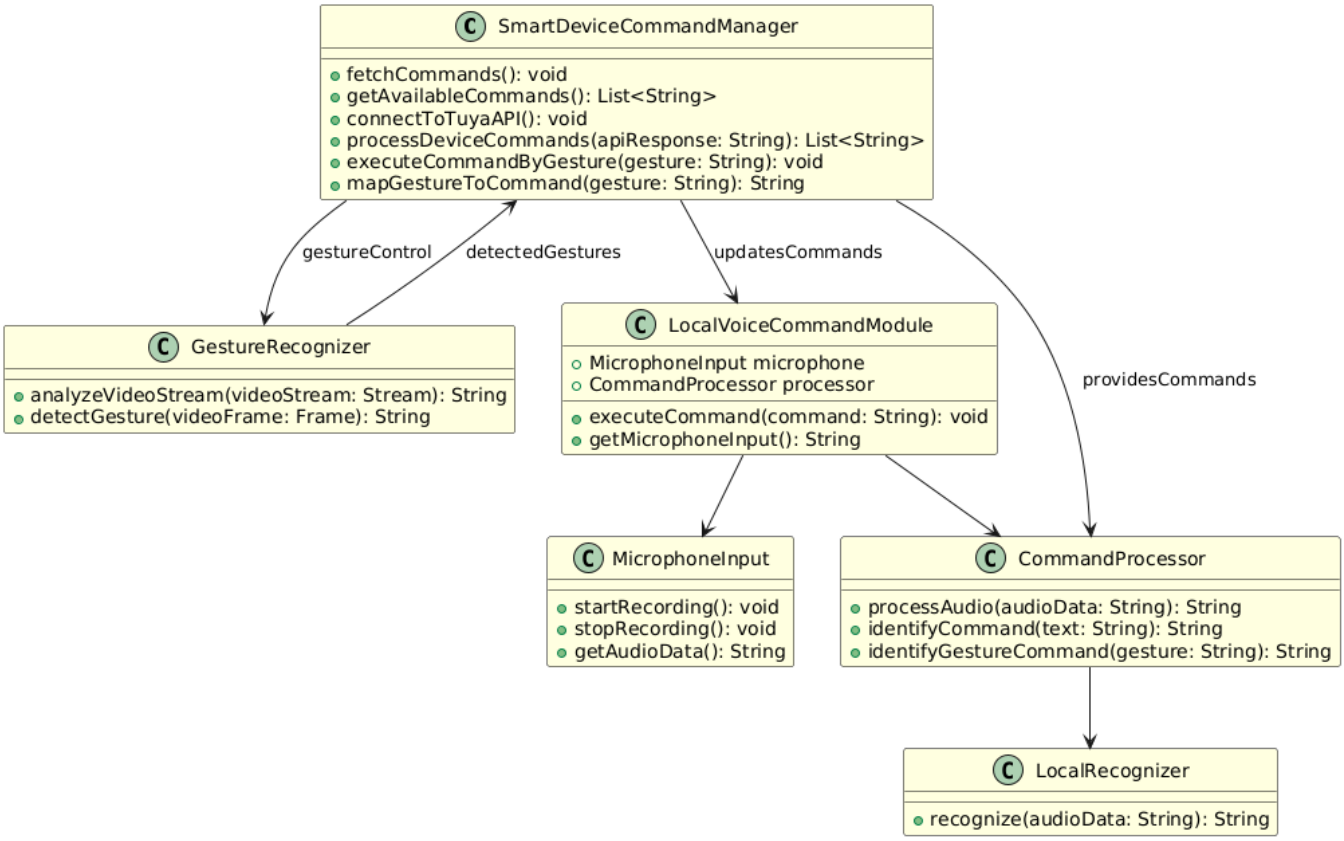


Рисунок 2.3 – Розширена діаграма класів

Клас GestureRecognizer відповідає за аналіз відеопотоку та розпізнавання жестів:

– **analyzeVideoStream(videoStream: Stream)** аналізує потік відео з камер для виявлення жестів.

– detectGesture(videoFrame: Frame) розпізнає жести в окремих кадрах відео.

Взаємодія із системою

– GestureRecognizer передає розпізнані жести до SmartDeviceCommandManager.

– SmartDeviceCommandManager відправляє відповідні команди до смарт-пристроїв на основі зіставлення жестів з командами.

– Інтеграція з CommandProcessor дозволяє обробляти як голосові, так і жестові команди, забезпечуючи мультиканальне управління системою.

Ця модифікація дає змогу додати новий спосіб управління "розумним" будинком через жести, зберігаючи модульність і масштабованість системи.

2.3 Моделі інтеграції голосових команд та жестів

Як зазначено в попередньому розділі, голосовий користувацький інтерфейс може бути реалізований двома способами:

- функціональність вбудована в панель керування інтелектуальних домашніх систем, яка має захищене з'єднання з хмарною інфраструктурою;
- функціональність розташована в окремому шарі у структурі інтелектуальних домашніх систем.

Алгоритмі інтеграції голосових команд ілюструє взаємодію на основі голосу між користувачем і системою розумного дому як послідовність запитів і відповідей:

- користувач розумного дому вводить висловлювання (голосову команду) з певним значенням.
- пристрій користувача (комп’ютер, ноутбук, планшет, смартфон або носимий пристрій) записує мову та передає її як аудіопотік до функції розпізнавання.
- алгоритм отримує відповідь на свій запит, тобто текст, який відповідає переданим аудіоданим;
- алгоритм витягує значення вхідного висловлювання користувача та формує запит із отриманою командою для сервера розумного дому.
- функція отримує запит, перевіряє його автентичність і коректність, готує та надсилає команду до підсистеми розумного дому, для якої вона призначена;
- підсистема розумного дому отримує команду, виконує її та повертає відповідну відповідь;
- система готує текст відповіді та надсилає її;
- інтерфейс отримує результат свого запиту у вигляді аудіофайлу, та надсилає її до пристрою користувача;
- пристрій користувача відтворює аудіофайл, отриманий у результаті запиту, і користувач чує голосову відповідь системи на своє висловлювання.

Кожен структурний шаблон команди визначається спеціальним синтаксисом і містить леми слів, групи синонімів, слоти параметрів тощо. Він визначає допустимі

позиції ключових слів і слотів параметрів у висловлюванні. Він використовується як основа для виявлення відповідності між висловлюванням і шаблоном.

Кожен слот відповідає одному параметру, необхідному для команди. Для нього визначені конкретні запитання для запиту значення відповідного параметра у користувача.

Механізм генерації команд – це процедура, в якій обробляються отримані значення параметрів команди та генерується формальна команда для сервера. Він може містити інші процедури, посилання на зовнішні служби та URL-посилання.

Алгоритм виявлення голосової команди з висловлювання користувача:

1. Користувач вводить голосове висловлювання.

2. Усі вузли зі списку попередньо визначених вузлів застосунку проходяться послідовно. Процес починається з першого вузла у списку.

3. Для кожного вузла послідовно проходяться всі структурні шаблони, пов’язані з цим вузлом.

4. Висловлювання порівнюється з поточним структурним шаблоном.

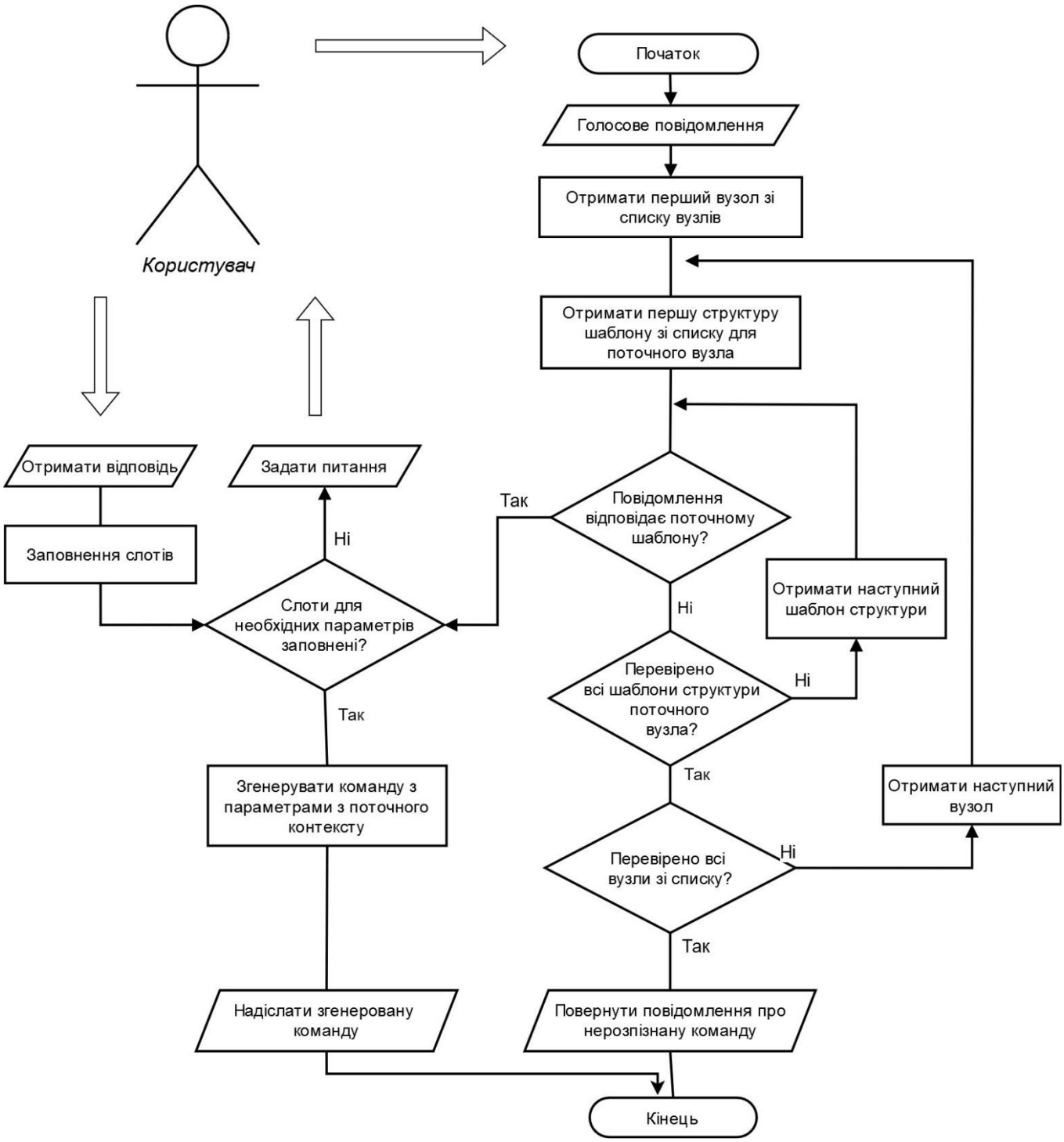
5. Якщо знайдено відповідність, проходження вузлів припиняється, і команда, що відповідає поточному вузлу, вважається розпізнаною.

6. Якщо команда розпізнана, перевіряється наявність усіх необхідних параметрів у висловлюванні або в поточному контексті.

а) Якщо в контексті все ще відсутні значення для всіх необхідних параметрів, користувачу задаються уточнювальні питання щодо кожного відсутнього значення, і після отримання відповідей ці значення додаються до контексту.

б) Якщо всі необхідні параметри мають відповідні значення в контексті, генерується команда, яка включає параметри, накопичені в контексті, і ця команда надсилається на пристрій виконання.

7. У разі відсутності відповідності після проходження всіх вузлів та їх структурних шаблонів генерується повідомлення про нерозпізнану команду, яке надсилається користувачу.



2.4 - Структура алгоритму виявлення голосової команди

Алгоритм на рисунку 2.5 ілюструє розпізнавання жестових команд, котре складається з наступних основних етапів:

1.Отримання та попередня обробка відеопотоку**:**

- захоплення відео з CCTV камери;
- детекція людини в кадрі;
- визначення ключових точок скелету (pose estimation);
- відстеження положення рук та пальців (hand tracking).

2.Аналіз та розпізнавання жестів**:**

- послідовний перебір вузлів зі списку шаблонів;
- для кожного вузла аналіз всіх пов’язаних шаблонів жестів;
- порівняння послідовності рухів з шаблонами;
- визначення відповідності поточного жесту шаблону.

3. Обробка параметрів команди**:**

а) Перевірка наявності всіх необхідних параметрів

б) У разі відсутності параметрів:

- очікування додаткових жестів-уточнень;
- аналіз вказівних жестів;
- можлива активація мультимодальної взаємодії;
- отримання параметрів через голосові команди.

в) Додавання отриманих параметрів до контексту

3.Генерація та виконання команди:

- Створення формальної команди з параметрами;
- Відправка команди на пристрій виконання.

4.Обробка нерозпізнаних жестів:

- Показ повідомлення про нерозпізнаний жест;
- Відображення підказки з правильним виконанням;
- Оновлення історії взаємодії;
- Збереження контексту команд.

5.Контекстна обробка:

- Підтримка історії взаємодії;
- Збереження стану системи;
- Врахування попередніх команд;
- Запобігання випадковим спрацюванням

Рисунок відображає цей процес з розділенням на логічні доріжки для кращого розуміння відповідальності кожного компонента системи.

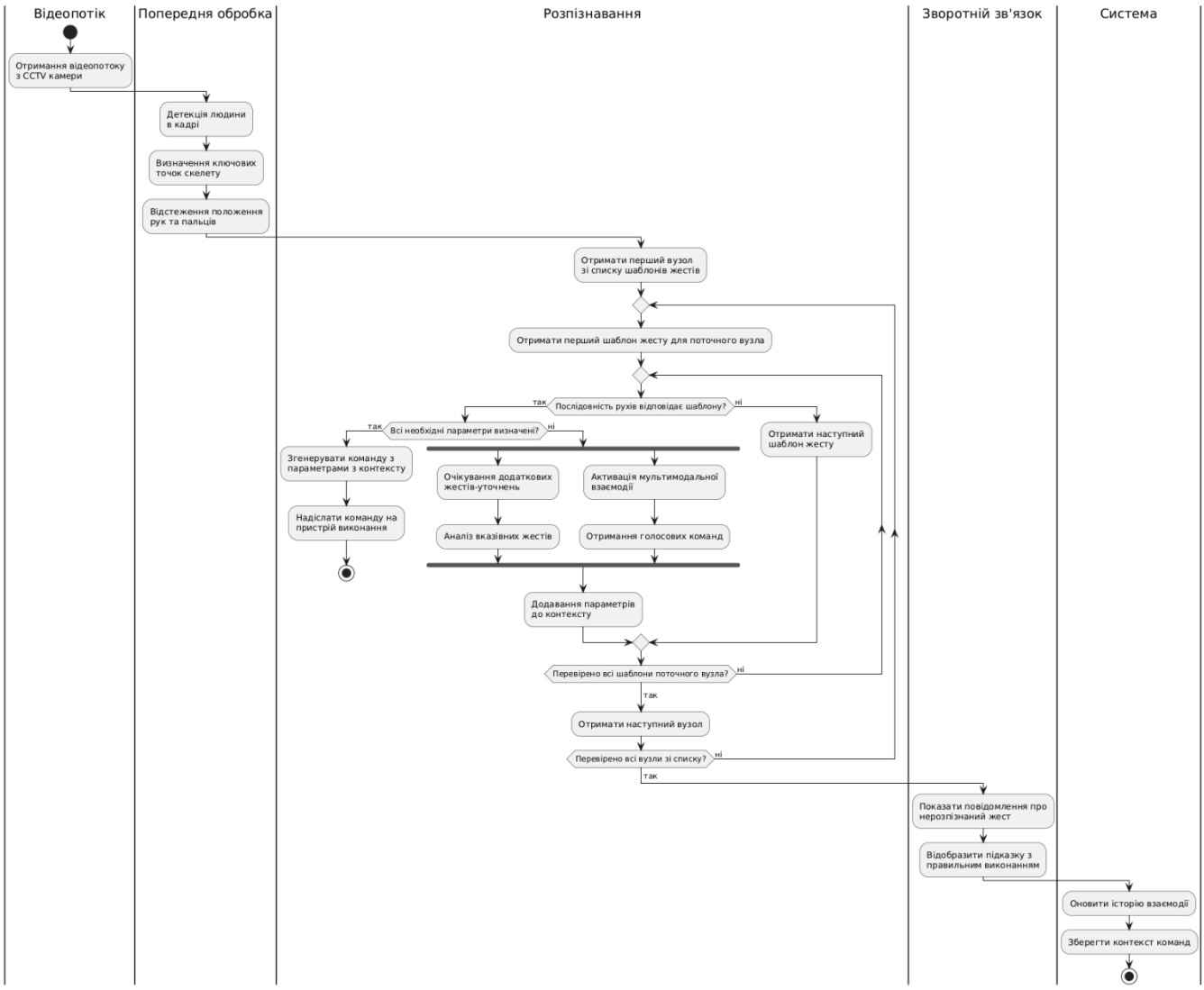


Рисунок 2.5 – Комбінована структура алгоритму розпізнавання жестових команд

Критерії ефективності кожного етапу алгоритму розпізнавання жестових команд можуть бути визначені відповідно до специфіки виконуваних завдань на кожному кроці. Ефективність оцінюється за швидкістю, точністю, стійкістю до шумів, адаптивністю до умов середовища та інтеграцією з іншими системами.

На етапі отримання та попередньої обробки відеопотоку критеріями ефективності є швидкість обробки відео в реальному часі, точність детекції людини в кадрі та визначення ключових точок скелету, а також стабільність алгоритмів відстеження положення рук і пальців. Точність визначення ключових точок значно впливає на подальші етапи аналізу жестів, тому важливо враховувати чутливість алгоритмів до різних умов освітлення, роздільної здатності камери та рівня шуму в відеопотоці.

На етапі аналізу та розпізнавання жестів ключовими критеріями є точність ідентифікації жесту відповідно до заданих шаблонів та швидкість порівняння послідовностей рухів із бібліотекою жестів. Ефективність цього етапу також визначається здатністю алгоритму правильно інтерпретувати складні жести, що складаються з кількох послідовних рухів, а також відсутністю помилкових спрацювань через випадкові рухи.

Етап обробки параметрів команди оцінюється за здатністю алгоритму коректно перевіряти наявність необхідних параметрів і додавати нові параметри до контексту. Ефективність цього етапу залежить від швидкості та точності обробки додаткових жестів або голосових команд, а також від інтеграції мультимодальної взаємодії для уточнення параметрів. Важливим критерієм є мінімізація часу, необхідного для отримання повного набору параметрів.

На етапі генерації та виконання команди ефективність визначається точністю формування команди, її відповідністю початковому жесту та швидкістю передачі команди на пристрій виконання. Забезпечення надійності передачі команди є критичним, особливо для систем, які працюють у реальному часі.

Етап обробки нерозпізнаних жестів оцінюється за якістю зворотного зв’язку, наданого користувачеві. До критеріїв входять інформативність повідомлення про помилку, чіткість підказки для правильного виконання жесту, а також здатність системи оновлювати історію взаємодії для навчання або вдосконалення алгоритму.

Контекстна обробка визначається здатністю системи ефективно зберігати історію взаємодії, враховувати попередні команди та стан системи для уникнення конфліктів або випадкових спрацювань. Важливим критерієм є забезпечення когерентності системної відповіді та мінімізація помилкових дій унаслідок неповного або некоректного контексту.

Локальні бібліотеки, що вирішують завдання визначення ключових точок скелету (pose estimation) та відстеження положення рук і пальців (hand tracking), є важливими компонентами сучасних систем аналізу жестів та рухів людини. До таких бібліотек належать OpenPose [13], MediaPipe Pose [14] та DeepLabCut [15].

OpenPose є потужною бібліотекою для аналізу рухів, яка забезпечує визначення до 135 ключових точок на людському тілі, включаючи тіло, обличчя,

руки та ноги. Ця бібліотека підтримує обробку відеопотоку в реальному часі та дозволяє працювати з кількома об'єктами одночасно. Вона використовує попередньо навчені моделі, що гарантують високу точність визначення положення точок скелету. Однак OpenPose має значні вимоги до ресурсів, особливо при використанні графічних процесорів, що обмежує її застосування на пристроях із низькою обчислювальною потужністю.

MediaPipe Pose, розроблена компанією Google, є легкою бібліотекою, яка дозволяє визначати 33 ключові точки на тілі людини. Ця бібліотека оптимізована для роботи як на CPU, так і на GPU, що забезпечує її продуктивність навіть на мобільних пристроях. MediaPipe Pose інтегрується через зручний API і підтримує обробку відеопотоку в реальному часі. Завдяки цьому вона є популярним вибором для мобільних додатків і проектів, які потребують швидкого виконання алгоритмів у реальному часі. Проте у складних сценах її точність може бути нижчою порівняно з OpenPose.

DeepLabCut є інструментом, орієнтованим на наукові дослідження, зокрема для аналізу рухів у різноманітних експериментальних умовах. Ця бібліотека використовує методи глибокого навчання для налаштування моделей, які спеціалізуються на розпізнаванні специфічних рухів. Вона забезпечує високу точність і гнучкість у виборі ключових точок, однак вимагає попереднього навчання моделей, що може бути складним і ресурсозатратним процесом.

Рисунок 2.6 - Розпізнавання жестів

Для відстеження положення рук і пальців найбільш поширеними рішеннями є MediaPipe Hands і OpenPose з відповідними модулями. MediaPipe Hands забезпечує ефективний аналіз рухів рук, визначаючи 21 ключову точку на кожній руці. Бібліотека відрізняється швидкістю роботи та оптимізацією для мобільних пристроїв, що робить її придатною для інтерактивних застосунків. OpenPose, у свою чергу, пропонує детальніше розпізнавання положення пальців, але має вищі вимоги до обчислювальних ресурсів.

Вибір відповідної бібліотеки залежить від конкретних вимог до точності, продуктивності та ресурсів системи, а також від умов, у яких передбачається її використання. Ці бібліотеки забезпечують широкі можливості для створення систем аналізу рухів та жестів, зокрема в контексті розумних середовищ і інтерактивних інтерфейсів.

2.4 Висновки до розділу 2

Архітектура системи інтеграції жестів та голосу побудована на принципах модульності, що сприяє чіткому розподілу функціональних завдань та забезпечує високу зручність у її модернізації та масштабуванні. Кожен модуль функціонує автономно, що дозволяє проводити заміну або оновлення окремих компонентів без впливу на роботу інших частин системи. Такий підхід сприяє поступовій розробці системи, полегшує тестування окремих функціональних блоків і зменшує ризик виникнення складних взаємозалежностей між її елементами.

Ефективність алгоритмів розпізнавання жестових команд залежить від балансу між швидкістю, точністю та адаптивністю на кожному з етапів, що забезпечує високу функціональність системи та зручність для користувачів.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ АЛГОРИТМІВ

3.1 Розроблення архітектури програмного засобу

Багаторівневу діаграму архітектури розумного будинку, яка включає наступні рівні [19-25].

1. Рівень пристроїв:
   - сенсори для збору даних;
   - виконавчі пристрої для керування;
   - локальні контролери для базової автоматизації.
2. Рівень шлюзів:
   - домашній шлюз для локального керування;
   - локальна база даних для зберігання критичних даних.
3. Рівень комунікації:
   - MQTT брокер для асинхронного обміну повідомленнями;
   - REST API для синхронної взаємодії;
   - WebSocket сервер для реального часу.
4. Хмарний рівень:
   - хмарне сховище для довготермінового зберігання даних;
   - сервіси обчислень для складної аналітики;
   - хмарна база даних для масштабованого зберігання.
5. Рівень додатків:
   - мобільний додаток для користувачів;
   - веб-інтерфейс для налаштування;
   - аналітична панель для моніторингу.

Між рівнями показані основні протоколи зв'язку (WiFi, Zigbee, MQTT, HTTP, WebSocket (рисунок 3.1). Така архітектура забезпечує:

- локальну роботу при відсутності інтернету;
- масштабованість через хмарні сервіси;
- гнучкість у виборі протоколів комунікації;
- різні інтерфейси для різних потреб користувачів.

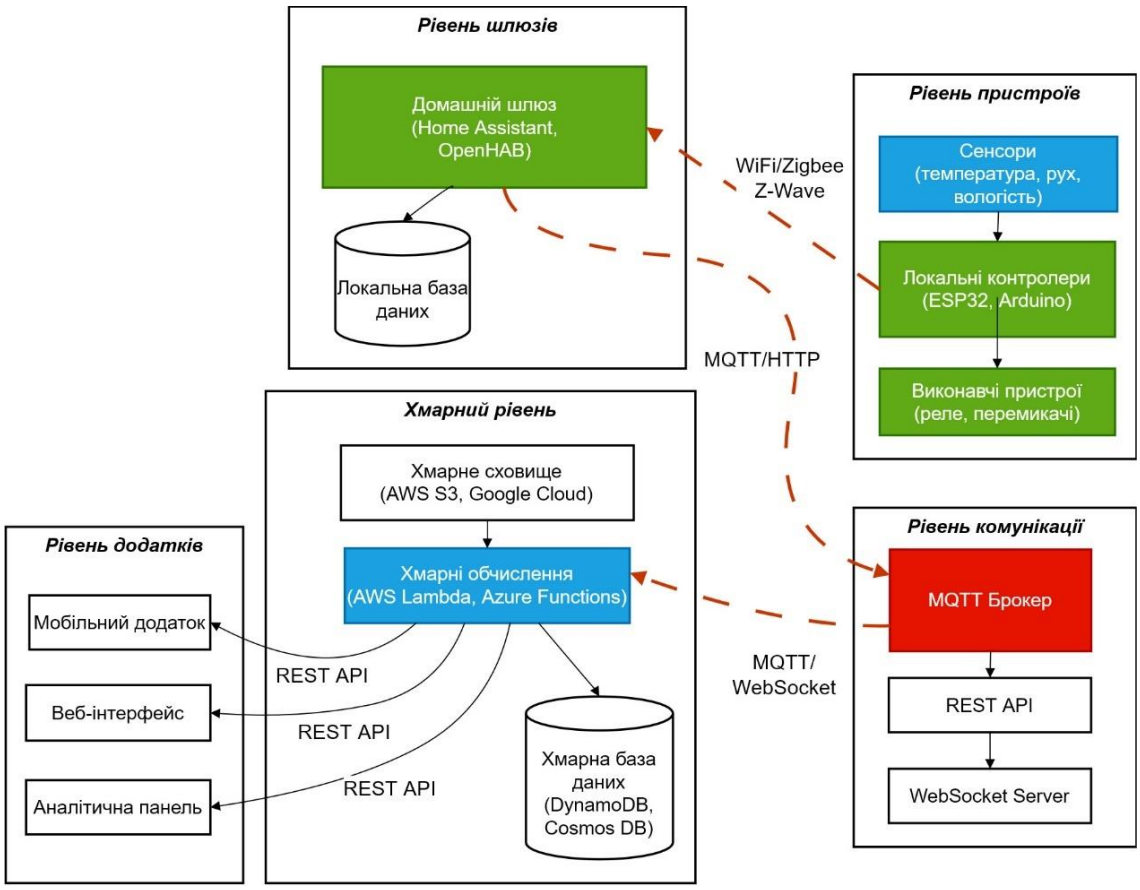


Рисунок 3.1 – Рівнева архітектура системи

Рівень шлюзів (Gateway Layer) є критично важливим компонентом в архітектурі розумного будинку, оскільки він служить мостом між локальними пристроями та зовнішніми системами (Рисунок 3.2).

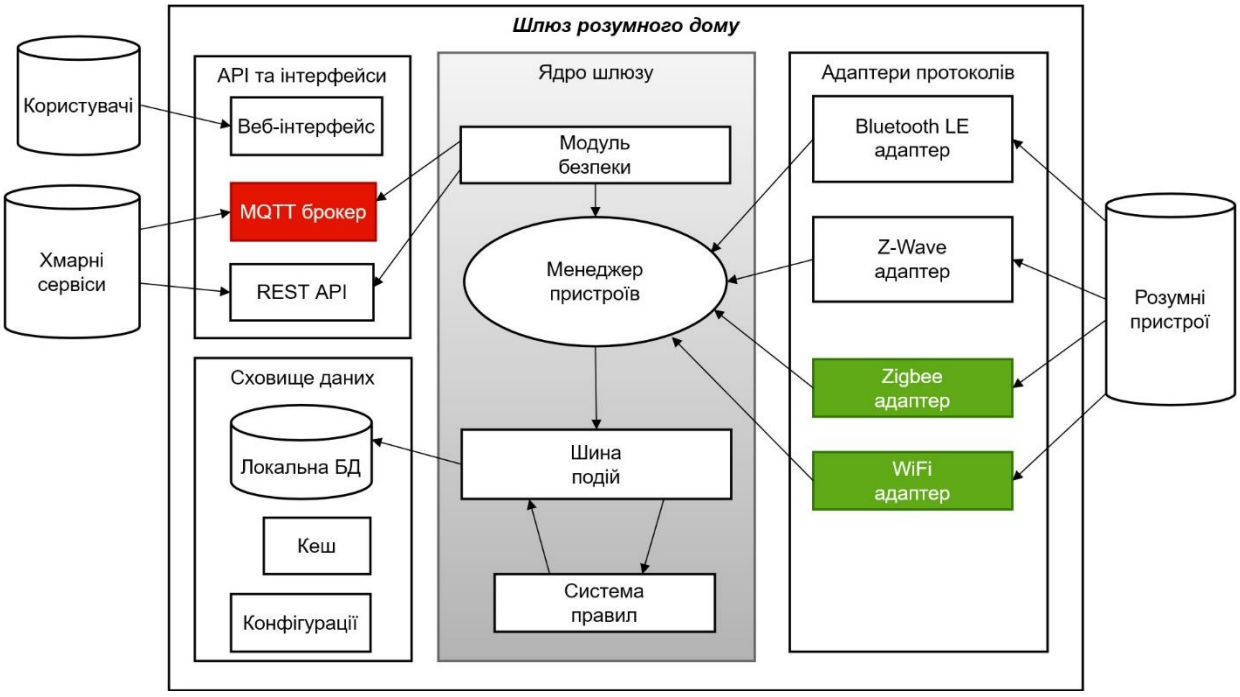


Рисунок 3.2 – Рівень шлюзів (Gateway Layer)

Основні компоненти шлюзів для інтелектуальних систем включають адаптери протоколів, ядро шлюзу, сховище даних та API з інтерфейсами. Адаптери протоколів забезпечують підтримку різних стандартів комунікації, таких як WiFi, Zigbee, Z-Wave та Bluetooth LE, виконуючи функцію перекладу між протоколами та управління фізичним підключенням пристроїв. Ядро шлюзу включає менеджер пристроїв для їх реєстрації та управління, систему правил для автоматизації та реалізації логіки прийняття рішень, шину подій для асинхронної взаємодії між компонентами та модуль безпеки для автентифікації й авторизації.

Сховище даних шлюзу складається з локальної бази даних, яка використовується для зберігання історичних даних та станів пристроїв, кешу для швидкого доступу до часто запитуваної інформації, а також конфігураційних файлів для налаштувань системи та пристроїв. API та інтерфейси шлюзу включають REST API для інтеграції з зовнішніми системами, MQTT брокер для організації асинхронної комунікації та веб-інтерфейс для локального управління.

Серед найпопулярніших реалізацій шлюзів варто виділити такі платформи, як Home Assistant**,** що є відкритою системою з широкою підтримкою пристроїв, OpenHAB**,** яка пропонує гнучкість та сумісність із різними протоколами, Apple HomeKit**,** що функціонує в закритій екосистемі з підвищеним рівнем безпеки, та Samsung SmartThings**,** орієнтована на хмарні рішення з широкими можливостями інтеграції.

Безпека на рівні шлюзів є критичним аспектом їх функціонування. Основні заходи включають шифрування всіх комунікацій, забезпечення локальної автентифікації користувачів, ізоляцію мереж пристроїв та регулярне оновлення систем безпеки. Ці заходи дозволяють мінімізувати ризики несанкціонованого доступу та забезпечують стабільну роботу шлюзів у мережі інтелектуальних систем.

Home Assistant, OpenHAB, Apple HomeKit та Samsung SmartThings є популярними платформами для інтеграції і управління розумними домашніми пристроями, включаючи розширення функціональності за допомогою плагінів для Tuya, а також модулів для голосового управління і управління жестами. Розглянемо особливості кожної з них з точки зору можливостей розширення.

Платформа Home Assistant [3] пропонує потужну інтеграцію з Tuya, включаючи підтримку локального режиму, що мінімізує залежність від хмарних сервісів. Ця платформа відзначається активною спільнотою, яка забезпечує швидке оновлення плагінів для підтримки нових пристроїв Tuya. Водночас, налаштування інтеграцій потребує технічних знань, зокрема роботи з API та конфігураційними файлами. Для розширення функціоналу голосових команд та управління жестами Home Assistant забезпечує легку інтеграцію з такими голосовими асистентами, як Google Assistant та Amazon Alexa, а також підтримує використання жестових інтерфейсів через такі пристрої, як Leap Motion чи OpenCV. Відкрита архітектура цієї платформи дозволяє розробляти власні модулі за допомогою Python або Node-RED, що надає значну гнучкість.

OpenHAB, [4] у свою чергу, забезпечує інтеграцію з Tuya за допомогою специфічних модулів, таких як MQTT або Tuya-MQTT Bridge. Хоча ця платформа підтримує широкий спектр пристроїв, процес інтеграції є менш інтуїтивним і може вимагати налаштування додаткових серверів. Розширення функціоналу голосових команд у OpenHAB реалізується через зовнішні доповнення, а інтеграція управління жестами можлива за допомогою апаратних рішень, таких як Arduino чи Raspberry Pi. Однак, складність створення нових модулів дещо перевищує рівень Home Assistant, що робить цю платформу менш доступною для початківців.

Apple HomeKit [16] орієнтована на інтеграцію з екосистемою Apple і забезпечує високий рівень автоматизації та зручності. Інтеграція з Tuya можлива лише через сторонні інструменти, такі як Homebridge, що вимагає технічних знань і не забезпечує повної підтримки локального управління. Голосові команди реалізуються через Siri, що інтегрована в екосистему Apple, проте можливості розширення функціоналу обмежені. Управління жестами обмежується базовою підтримкою через пристрої Apple, але можливості створення власних модулів у цій сфері є мінімальними через закритість екосистеми.

Samsung SmartThings [17] забезпечує офіційну підтримку Tuya через інтеграцію API, що дозволяє автоматичне розпізнавання пристроїв. Ця платформа є простою в налаштуванні, але залежить від хмарних сервісів Tuya, що обмежує її гнучкість. Голосове управління реалізується через інтеграцію з Bixby, Google

Assistant та Amazon Alexa, а управління жестами доступне через підтримувані пристрої, наприклад камери Samsung. Водночас розширення функціоналу через створення власних модулів є складним через політику закритої екосистеми. В таблиці 3.1 порівняємо платформи.

Таблиця 3.1 - Порівняння платформ

Платформа	Tuya інтеграція	Голосові команди	Жести	Простота розширення
Home Assistant	Відмінна (локальний режим)	Широка підтримка	Можливість створення модулів	Найвища
OpenHAB	Гнучка, але складна	Модульна інтеграція	Підтримка зовнішніх пристроїв	Середня
Apple HomeKit	Лише через сторонні інструменти	Вбудована Siri	Обмежена підтримка жестів	Низька
Samsung SmartThings	Офіційна, але залежить від хмари	Підтримка Bixby, Alexa, Google	Часткова підтримка через пристрої	Низька

Таким чином, Home Assistant пропонує найширші можливості для розширення функціональності за рахунок відкритої архітектури і гнучкої інтеграції з різними технологіями. OpenHAB забезпечує значний рівень гнучкості, проте вимагає більшого технічного досвіду. Apple HomeKit відзначається високою зручністю для користувачів екосистеми Apple, але має суттєві обмеження у налаштуванні. Samsung SmartThings пропонує просте і зручне налаштування, але закрита архітектура платформи обмежує можливості її розширення.

Пристрої Tuya [18], які підключаються до Home Assistant, мають певну структуру, що забезпечує їх взаємодію через API або локальну мережу. Ця структура містить кілька ключових компонентів, кожен з яких відповідає за специфічні функції. Основні компоненти пристрою:

– Мікроконтролер Tuya - основна мікросхема, яка забезпечує обробку команд і управління функціями пристрою. Він відповідає за комунікацію через WiFi, Zigbee або Bluetooth;

– Інтерфейс передачі даних забезпечує зв'язок із зовнішніми платформами, зокрема через MQTT або Tuya Cloud API;

– Сенсори/Актори - фізичні компоненти пристрою, такі як вимикачі, реле, датчики температури, освітлення тощо.

Пристрій Tuya зазвичай виконує одну або кілька функцій, які відповідають його типу. Основними є:

– датчики - вимірюють фізичні величини, такі як температура, вологість, рівень освітлення або рух.

– актори - Виконують активні дії, наприклад, увімкнення/вимкнення реле, зміна яскравості лампи чи відкриття/закриття жалюзі.

– керування - Забезпечує підтримку сценаріїв автоматизації, таймерів, або ручного управління.

Команди, які підтримуються пристроями Tuya, поділяються на кілька категорій залежно від їх функцій. Команди управління станом:

– power – вмикання/вимикання пристрою;

– brightness – налаштування яскравості (у відсотках);

– temperature – зміна температури (в градусах);

– mode – вибір режиму роботи (наприклад, «Авто», «Ручний»).

Команди Отримання даних:

– status – поточний стан пристрою.

– sensor_value – значення, зчитане з датчика (наприклад, температура).

Системні команди:

– reset – скидання налаштувань пристрою;

– update – запит на оновлення прошивки.

Комунікація з пристроями Tuya в Home Assistant зазвичай здійснюється через JSON-формат. Кожна команда має структуру, яка включає ключі та значення:

– dp_id (Data Point ID): Унікальний ідентифікатор функції або параметра.

– value: Значення, яке передається або зчитується.

Приклад структури команди JSON:

```
{ "dp_id": 1,
  "value": true }
```

3.2 Програмна реалізація алгоритмів

Реалізація плагіна для Home Assistant**,** який включає клас `SmartDeviceCommandManager` для взаємодії зі смарт-пристроями Tuya. Код написаний на Python, враховуючи стандартні вимоги до інтеграцій у Home Assistant.

```
import logging
import requests
_LOGGER = logging.getLogger(__name__)
class SmartDeviceCommandManager:
    def __init__(self, tuya_api_url, api_key, api_secret):
        # Ініціалізація для роботи зі смарт-пристроями Tuya
        self.tuya_api_url = tuya_api_url  # URL API Tuya
        self.api_key = api_key  # Ключ доступу до API
        self.api_secret = api_secret  # Секретний ключ API
        self.commands = []  # Список доступних команд
```

Функція **fetchCommands** отримує список команд із API Tuya. Вона викликає connectToTuyaAPI для підключення до сервісу та використовує processDeviceCommands для обробки відповіді. Якщо виникає помилка, вона логікує її у файл журналу.

```
    def fetchCommands(self):
        """Отримує список команд із API Tuya та обробляє їх."""
        _LOGGER.info("Отримання команд із API Tuya...")
        try:
            response = self.connectToTuyaAPI()  # Підключення до API
            self.commands = self.processDeviceCommands(response)
        except Exception as e:
            _LOGGER.error(f"Помилка під час отримання команд: {e}")
```

Функція **getAvailableCommands** Повертає список доступних команд, які були завантажені та оброблені раніше.

```
    def getAvailableCommands(self):
        """Повертає список доступних команд."""
        return self.commands
```

Функція connectToTuyaAPI Виконує HTTP-запит до API Tuya для отримання команд. Вона налаштовує заголовки, включаючи ключ API та підпис, перевіряє відповідь сервера та повертає результат.

```
    def connectToTuyaAPI(self):
        """Підкл до API Tuya та отрим список команд пристрою."""
        _LOGGER.info("Підключення до API Tuya...")
        headers = {
            "Content-Type": "application/json",
            "client_id": self.api_key,
            "sign": self._generate_api_signature(),
        }
```

Перевірка статусу відповіді

```
        try:
            response = requests.get(f"{self.tuya_api_url}/commands",
headers=headers)
            response.raise_for_status()
            _LOGGER.debug(f"Відповідь API: {response.text}")
            return response.text
        except requests.RequestException as e:
            _LOGGER.error(f"Помилка підключення до API Tuya: {e}")
            raise
```

Функція **processDeviceCommands** Парсить JSON-відповідь від API Tuya, витягує список доступних команд та повертає його. Якщо відповідь некоректна, функція повертає порожній список і логікує помилку.

```
    def processDeviceCommands(self, apiResponse):
        """Парсить відповідь API та витягує доступні команди."""
        _LOGGER.info("Обробка відповіді API для отримання команд
пристроїв...")
        try:
            response_data = requests.json.loads(apiResponse)  # Парсинг
            commands = response_data.get("commands", [])  #Отрим команд
            _LOGGER.debug(f"Отримані команди: {commands}")
            return commands
        except Exception as e:
            _LOGGER.error(f"Помилка під час обробки відповіді API:
{e}")
            return []
```

Функція executeCommandByGesture Приймає жест, мапить його на відповідну команду через mapGestureToCommand, а потім викликає _sendCommandToDevice, щоб виконати команду.

```
    def executeCommandByGesture(self, gesture):
        """Мапить жест на команду та виконує її."""
        _LOGGER.info(f"Виконання команди для жесту: {gesture}")
        command = self.mapGestureToCommand(gesture)    # Визначення
команди для жесту
        if command:
            self._sendCommandToDevice(command)  # Надсилання команди
        else:
            _LOGGER.warning(f"Для жесту {gesture} не знайдено
відповідної команди.")
```

Функція mapGestureToCommand мапить жести (наприклад, "swipe_up", "wave_left") на команди (наприклад, "increase_brightness", "turn_off"). Якщо жест не має відповідної команди, повертається None.

```
    def mapGestureToCommand(self, gesture):
        """Мапить жест на відповідну команду."""
        gesture_command_map = {
            # Жест підвищення яскравості
            "swipe_up": "increase_brightness",
            # Жест зниження яскравості
            "swipe_down": "decrease_brightness",
            "wave_left": "turn_off",  # Жест вимкнення пристрою
            "wave_right": "turn_on",  # Жест увімкнення пристрою
        }
        return gesture_command_map.get(gesture, None)
```

Функція _sendCommandToDevice надсилає конкретну команду на пристрій через API Tuya. Формує структуру запиту, виконує HTTP-запит та перевіряє результат. Якщо виникає помилка, вона логікується.

```
    def _sendCommandToDevice(self, command):
        """Надсилає конкретну команду на пристрій Tuya."""
        _LOGGER.info(f"Надсилання команди пристрою: {command}")
        # Структура запиту
        payload = {"command": command}
        headers = {
            "Content-Type": "application/json",
            "client_id": self.api_key,
            "sign": self._generate_api_signature(),
```

```
        }

        try:
            response  =  requests.post(f"{self.tuya_api_url}/execute",
json=payload, headers=headers)
            response.raise_for_status()  # Перевірка статусу відповіді
            _LOGGER.info(f"Команду успішно виконано: {command}")
        except requests.RequestException as e:
            _LOGGER.error(f"Помилка надсилання команди пристрою: {e}")
```

Функція _generate_api_signature генерує підпис для аутентифікації в API Tuya. Реалізація залежить від специфікації API Tuya.

```
    def _generate_api_signature(self):
        """Генерує підпис для аутентифікації в API Tuya."""
        # код для генерації підпису ...
        return "generated_signature"
```

Таблиця 3.2 давачів смарт-будинку, їх основні параметри, імовірні команди, а також приклади голосових команд і жестів керування [26-35].

Таблиця 3.2 - Давачі «розумного» будинку

Давач	Основні параметри	Команди	Приклад голосової команди	Приклад жесту керування
1	2	3	4	5
Давач температури	Температура (°C), вологість (%)	Запитати значення, встановити поріг температури	"Яка температура в спальні?"	Підняття руки вгору для збільшення
Давач руху	Наявність руху, рівень освітлення (люкси)	Включити/вимкнути світло, активувати сигналізацію	"Увімкни світло у вітальні"	Помах рукою вбік
Датчик якості повітря	Рівень CO2 (ppm), вологість (%), температура (°C)	Запитати рівень CO2, увімкнути очищувач повітря	"Який рівень CO2 у кімнаті?"	Обертальний рух долоні
Датчик відкриття дверей	Стан (відкрито/ закрито)	Відкрити/закрити двері, активувати сигналізацію	"Закрий вхідні двері"	Змах долонею донизу

Продовження таблиці 3.2

1	2	3	4	5
Сенсор освітлення	Рівень освітлення (люкси)	Включити/вимкнути освітлення, змінити яскравість	"Зменши яскравість у вітальні"	Розкриття пальців у напрямку вгору
Датчик витоку води	Стан (сухо/волога), місце витоку	Закрити воду, активувати аварійний сигнал	"Перевір стан водопостачання"	Стиснення долоні у кулак
Розумний термостат	Температура (°C), режим роботи	Змінити температуру, змінити режим роботи	"Встанови температуру 22 градуси"	Рух долонею вгору або вниз
Сенсор диму	Рівень задимленості (ppm)	Активувати тривогу, скинути тривогу	"Скинути сигнал тривоги"	Плавне проведення рукою перед сенсором
Датчик тиску води	Тиск (бар)	Відкрити/закрити клапан, перевірити стан	"Який тиск у системі водопостачання?"	Круговий рух пальцем
Датчик звуку	Рівень шуму (дБ)	Зменшити шум, увімкнути білий шум	"Увімкни білий шум у спальні"	Піднесення пальця до рота

Розглянемо виконавчі пристрої в таблиці 3.3

Таблиця 3.3 - Виконавчі пристрої «розумного» будинку

Пристрій	Основні параметри	Імовірні команди	Приклад голосової команди	Приклад жесту керування
1	2	3	4	5
Розумна лампа	Яскравість (%), колір, режим роботи	Увімкнути/вимкнути, змінити яскравість, змінити колір	"Увімкни світло у вітальні"	Помах рукою вгору для збільшення яскравості
Розумна розетка	Стан (увімкнено/вимкнено), споживання енергії	Увімкнути/вимкнути, перевірити споживання	"Увімкни розетку на кухні"	Круговий рух долоні
Розумний замок	Стан (відкрито/закрито), статус батареї	Відкрити/закрити, перевірити стан	"Закрий двері у вхідній зоні"	Піднесення долоні до пристрою

Продовження таблиці 3.3

1	2	3	4	5
Розумні жалюзі	Стан (відкрито/закрито), рівень відкриття (%)	Відкрити/закрити, змінити рівень відкриття	"Відкрий жалюзі у спальні на 50%"	Розкриття пальців у напрямку вгору
Термостат	Температура (°C), режим (авто/ручний), поточний стан	Встановити температуру, змінити режим	"Встанови температуру на 22 градуси"	Рух долонею вгору або вниз
Система поливу	Час увімкнення, зона, тривалість	Увімкнути/вимкнути, налаштувати графік	"Увімкни полив у саду на 10 хвилин"	Помах рукою вбік
Розумний кондиціонер	Температура (°C), швидкість вентилятора	Увімкнути/вимкнути, встановити температуру	"Увімкни кондиціонер у вітальні на 24 градуси"	Рух долонею вниз
Розумний пилосос	Статус батареї, поточна локація	Увімкнути/вимкнути, відправити до док-станції	"Запусти прибирання у вітальні"	Круговий рух пальця
Розумний дзвінок	Відео/аудіо-зв'язок, стан батареї	Почати дзвінок, увімкнути/вимкнути мікрофон	"Відкрий камеру на вході"	Плавний рух рукою перед сенсором
Розумний телевізор	Гучність (%), канал, джерело сигналу	Увімкнути/вимкнути, змінити канал, налаштувати гучність	"Зміні канал на новини"	Змах рукою вліво чи вправо

3.3 Експериментальне дослідження алгоритмів

Діаграма послідовності (рисунок 3.3), яка показує три сценарії взаємодії користувача з системою розумного будинку:

1. чисто голосова команда (включення світла);
2. чисто жестова команда (підняття жалюзі);
3. комбінована команда з використанням жесту та голосу.

Діаграма показує:

- як сигнали від користувача проходять через різні системи розпізнавання;

- паралельну обробку голосу та жестів при комбінованих командах;
- зворотній зв'язок від пристроїв до користувача.

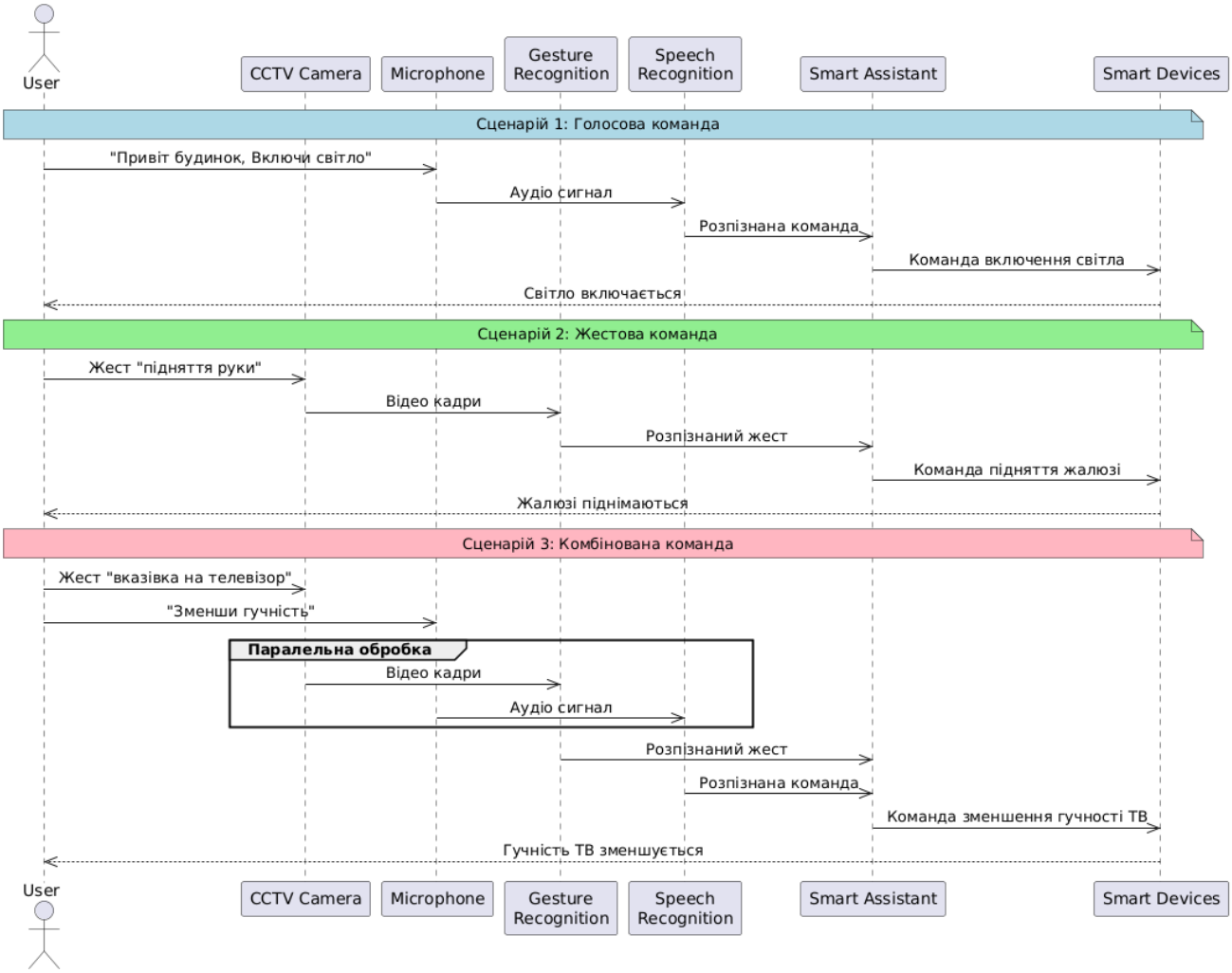


Рисунок 3.3 - Три сценарії взаємодії користувача з системою

Приклад роботи алгоритму розпізнавання голосових команд: намір користувача дізнатися температуру в розумному будинку (див розділ 2.3).

Користувач промовляє голосове висловлювання: "Яка температура в спальні?". Алгоритм починає послідовне виконання етапів.

На першому етапі користувацьке висловлювання фіксується пристроєм (наприклад, розумною колонкою або смартфоном) та передається у вигляді аудіопотоку для обробки.

Далі активується процес пошуку відповідності у вузлах застосунку. Усі вузли зі списку попередньо визначених вузлів системи обробляються послідовно, починаючи з першого. Наприклад, серед вузлів можуть бути: "управління освітленням", "контроль клімату", "перевірка безпеки". Висловлювання

порівнюється з шаблонами для кожного вузла. Для вузла "контроль клімату" система аналізує всі пов’язані структурні шаблони, такі як:

- "Яка температура?",
- "Що з температурою в [локація]?",
- "Дай температуру в [приміщення]".

Висловлювання користувача порівнюється з цими шаблонами. Знаходиться відповідність із шаблоном: "Що з температурою в [локація]?", де "спальня" розпізнається як значення для параметра [локація].

Після визначення відповідного шаблону система перевіряє, чи всі необхідні параметри присутні у висловлюванні або збережені в контексті. У цьому випадку параметр [локація] уже містить значення "спальня", тому додаткових уточнень від користувача не потрібно.

Система генерує формальну команду, яка включає параметри, отримані з контексту: "Запитати температуру в спальні". Ця команда надсилається на сервер або пристрій, відповідальний за збір даних з датчиків температури.

Відповідний пристрій (наприклад, термостат або кліматична станція) отримує команду, виконує її та надсилає відповідь: "Температура в спальні зараз 22°C".

Нарешті, система формує відповідь у вигляді тексту та аудіо, яке відтворюється користувачеві: "Температура в спальні 22 градуси за Цельсієм."

Якщо на етапі проходження вузлів або аналізу шаблонів не вдається знайти відповідність, користувач отримує повідомлення: "На жаль, я не зрозумів ваш запит. Спробуйте уточнити або повторити його." У випадку відсутності деяких параметрів система запитує: "Про яке приміщення йде мова?", щоб уточнити потрібну інформацію.

Тестовий приклад розпізнавання голосової команди і знаходження параметрів показано на рисунку 3.4.

Проведено тестування розроблених алгоритмів для сервера Home Assistant. Діаграма послідовності наочно показала, як сигнали користувача проходять через системи розпізнавання та обробляються паралельно при комбінованих командах, забезпечуючи зворотний зв’язок до користувача.

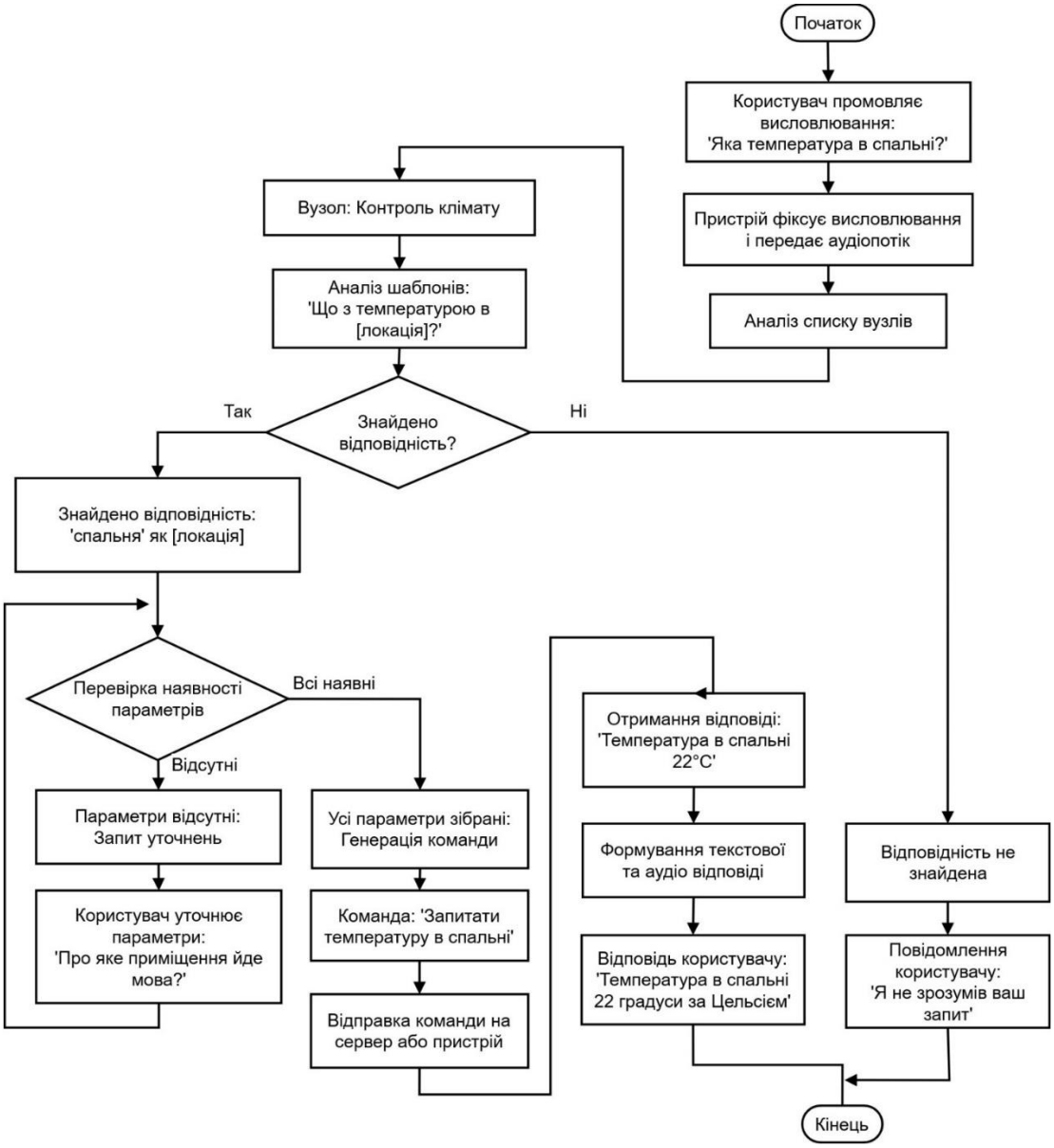


Рисунок 3.4 – Тестовий приклад розпізнавання голосової команди і знаходження параметрів

Результати експериментального дослідження відображені у вигляді таблиці, яка показує залежність точності розпізнавання жестів і голосу від відстані до камери/мікрофона. Також створено графік, який ілюструє зміну точності в цих умовах.

Для відстаней до камери від 1 до 10 метрів і відповідних розмірів користувача в кадрі (від 20% до 80%) отримані наступні результати. Точність розпізнавання жестів:

- при відстані 1 м (80% розмір у кадрі): точність близько 93%;
- при відстані 10 м (20% розмір у кадрі): точність знижується до ~30%;

– спостерігається суттєва залежність точності від відстані та розміру користувача в кадрі.

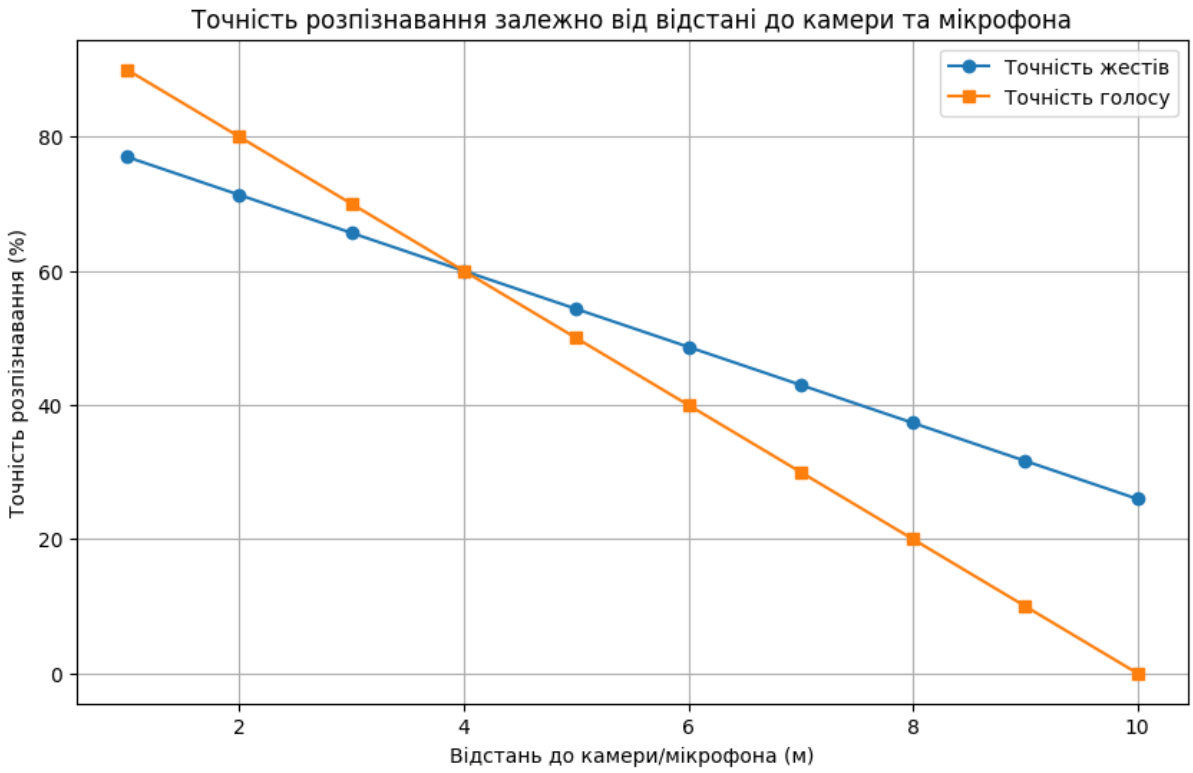


Рисунок 3.5 – Точність розпізнавання

Точність розпізнавання голосу:

– при відстані 1 м: точність 90%;

– при відстані 10 м: точність знижується до ~0%;

– точність голосових команд прямо залежить від відстані до мікрофона.

Ці результати свідчать, що збільшення відстані суттєво впливає на точність обох видів розпізнавання, але голосові команди деградують швидше.

Використання смартфона для введення голосових команд і жестів у системах управління «розумним» будинком має суттєві відмінності, які визначають їх ефективність та зручність залежно від конкретних умов. Смартфони оснащені високоякісними мікрофонами, що дозволяє забезпечувати високу точність розпізнавання голосу, особливо за умови використання сучасних алгоритмів шумозаглушення. Однак у шумних середовищах або за умов слабкого інтернет-з'єднання точність розпізнавання голосу може знижуватися, особливо якщо обробка виконується на хмарних сервісах. З іншого боку, локальні алгоритми розпізнавання

голосу, хоча й менш потужні, забезпечують більшу конфіденційність та автономність.

Жестове управління зі смартфонів ґрунтується на використанні вбудованих камер для аналізу рухів. У цьому аспекті смартфони поступаються спеціалізованим камерам, оскільки мають обмежену здатність до тривимірного відстеження жестів. Точність розпізнавання залежить від таких факторів, як якість освітлення, фон сцени та відстань користувача до камери. Крім того, для розпізнавання жестів камера смартфона повинна бути спрямована на користувача, що обмежує мобільність і вимагає додаткової взаємодії з пристроєм.

Обидва методи взаємодії мають різні обчислювальні вимоги. Голосове управління здебільшого покладається на хмарні обчислення, що знижує навантаження на смартфон, але залежить від доступу до інтернету. Натомість жестове управління часто реалізується локально, використовуючи алгоритми обробки зображень, такі як MediaPipe або OpenCV, що може суттєво навантажувати ресурси смартфона.

3.4 Висновки до розділу 3

Багаторівнева діаграма архітектури розумного будинку включає рівень пристроїв, рівень шлюзів, рівень комунікації, хмарний рівень, рівень додатків. Жести керування дають можливість безконтактного управління для додаткової зручності або у випадках, коли голосові команди недоречні (наприклад, у шумному середовищі). Наведено приклад, що ілюструє ефективну роботу алгоритму розпізнавання голосових команд з урахуванням контексту та забезпеченням зворотного зв'язку для користувача.

ВИСНОВКИ

1. Проведено аналіз методів управління системами «розумного» будинку, зокрема технологій розпізнавання голосу та жестів, протоколів інтеграції та архітектурних рішень. Використано сучасні методи машинного навчання, алгоритми розпізнавання ключових точок скелету для жестів і спектрального аналізу для голосових команд. Для інтеграція використовуються стандартні протоколи, такі як MQTT, HTTP і WebSockets, що забезпечило надійність і масштабованість.

2. Розроблено архітектуру системи управління «розумним» будинком, що інтегрує голосові команди та жести, із модульною структурою, яка включає обробку аудіосигналів, аналіз відеопотоку та зв'язок із пристроями через сучасні протоколи. Використано ефективні алгоритми розпізнавання голосу PocketSphinx та жестів MediaPipe, а також протокол MQTT для інтеграції з пристроями. Система здатна працювати з різними каналами взаємодії, включаючи голос і жести, що підвищує її адаптивність і забезпечує зручність управління «розумним» будинком навіть у складних умовах (наприклад, за відсутності інтернету).

3. Запропоновано комбіновані моделі інтеграції голосових команд і жестів, які дозволяють виконувати команди користувача через мультимодальне управління. Реалізовано алгоритми паралельної обробки голосу та жестів, що дозволяють системі синхронізувати результати різних модулів (розпізнавання тексту та ключових точок скелету). Розроблені моделі забезпечують точність і швидкість розпізнавання, мінімізацію помилок у визначенні команд, а також підтримку гнучких сценаріїв управління, таких як зміна параметрів за допомогою жестів або голосу.

4. Результати тестування алгоритму розпізнавання голосових команд показали ефективність системи у виконанні запитів, таких як отримання температури в конкретній кімнаті. Алгоритм успішно ідентифікував ключові параметри у висловлюванні користувача, обрав відповідний вузол системи , перевірив контекст на наявність усіх необхідних параметрів та сформував коректну команду для виконання. У випадках відсутності параметрів або помилок розпізнавання система забезпечувала користувача інформативними уточненнями або повідомленнями про неможливість виконання команди.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комар В.А. Інтеграція голосових команди та жестів для управління "розумним" будинком. Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «інтелектуальні комп'ютерні системи та мережі»5 листопада 2024.Тернопіль. С. 55

2. Куренчук А.С., Комар В.А. Сегментація зображень в автоматизованих системах з допомогою U-net мереж. Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «інтелектуальні комп'ютерні системи та мережі»5 листопада 2024.Тернопіль. С. 117

2. Заіка М. В. Дослідження кепстральних коефіцієнтів голосового сигналу користувача системи автентифікації. 2020 URL: https://openarchive.nure.ua/server/api/core/bitstreams/14aa88f4-0161-48a6-8a00-270e81950e90/content

3. Home Assistant URL: https://www.home-assistant.io

4. OpenHAB Empowering the smart home URL: https://www.openhab.org

5. MQTT: The Standard for IoT Messaging URL: https://mqtt.org

6. Hari Prabhat Gupta, Haresh S. Chudgar A Continuous Hand Gestures Recognition Technique for Human-Machine Interaction Using Accelerometer and Gyroscope Sensors URL: https://www.semanticscholar.org/paper/A-Continuous-Hand-Gestures-Recognition-Technique-Gupta-Chudgar/2742dd51ee37064eb71830a315d89650c988e4df

7. Yen-Cheng Chu Recognition of Hand Gesture Sequences by Accelerometers and Gyroscopes URL: https://www.mdpi.com/2076-3417/10/18/6507

8. Gyroscope-Based Continuous Human Hand Gesture Recognition for Multi-Modal Wearable Input Device for Human Machine Interaction URL: https://www.mdpi.com/1424-8220/19/11/2562

9. Marcos Negreiros Rylo Gesture recognition of wrist motion based on wearables sensors URL: https://www.sciencedirect.com/science/article/pii/S1877050922015927/pdf?md5=deb9948c86e9bf263e5ca6da4f1aaca9&pid=1-s2.0-S1877050922015927-main.pdf

10. Gesture Detection Using Accelerometer and Gyroscope URL: https://www.researchgate.net/publication/366232582_Gesture_Detection_Using_Accelerometer_and_Gyroscope

11. Titus, J.E., P. A., K., Job, M., & P., C. IoT Enabled Home Automation for Differently Challenged and Elderly People Using Gesture and Voice Recognition. 2022 Second International Conference on Next Generation Intelligent Systems (ICNGIS), 1-5. URL: https://doi.org/10.1109/ICNGIS54955.2022.10079753

12. WebSockets URL: https://datatracker.ietf.org/doc/html/rfc6455

13. OpenPose URL: https://github.com/CMU-Perceptual-Computing-Lab/openpose

14. MediaPipe Pose URL: https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/pose.md

15. DeepLabCut https://github.com/deeplabcut

16. Apple HomeKit URL: https://www.apple.com/home-app/

17 Samsung SmartThings URL: https://www.samsung.com/ua/smartthings/

18 Tuya Smart - Global Cloud Platform Service Provider URL: https://www.tuya.com

19. Бібліотека QueueSim. URL: https://github.com/A-Herzog/QueueSim.

20. Бібліотека Ciw. URL: https://ciw.readthedocs.io/en/latest/.

21. Бібліотека SimPy. URL: https://simpy.readthedocs.io/.

22. Модель телекомунікаційної мережі "інтелектуального будинку" / В. М. Теслюк та ін. *Науковий вісник НЛТУ України*. 2016. Т. 26.1. С. 351-357. URL: http://nbuv.gov.ua/UJRN/nvnltu_2016_26.1_56.

23. Використання XML для систем автоматизованого генерування моделей на основі мереж Петрі / В. М. Теслюк та ін. *Моделювання та інформаційні технології*. 2013. Т. 70. С. 129-136. URL: http://nbuv.gov.ua/UJRN/Mtit_2013_70_21.

24. Ткаченко В. П., Огірко І. В., Огірко О. І. Інформаційна модель Грід технології. *Системи обробки інформації*. 2017. Т. 4(150). С. 88—91.

25. Данченков Я.В. Теорія інформації: Навчальний посібник. Рівне: НУВГП; 2012. 111 с.

26. Сорока Л.С. Основи теорії інформаці: навчальний посібник. Харків: ХНУ ім. В.Н. Каразіна; 2007. 264 с.

27. Інформаційні матеріали сайту по Bluetooth. URL: http://www.bluetoothweb.com

28. Інформаційні матеріали сайту по Bluetooth. URL: http://www.bluetoothsig.org

29. Інформаційні матеріали сайту по IEEE 802.11 URL: http://www.80211-planet.com/news/

30. Інформаційні матеріали сайту по HIPERLAN2 URL: http://www.hiperlan2.com

31. Інформаційні матеріали сайту Wi-Fi Alliance. URL: http://www.wi-fi.org

32. Інформаційні матеріали сайту по Wi-Fi (точки доступа). URL: http://www.wi-fizone.org

33. Сайко В.Г., Казіміренко В.Я., Літвінов Ю.М. Мережі бездротового широкосмугового доступу. Навчальний посібник. К., ДУТ, 2015. 196 с.

34. Тимченко О.В., Зеляновський М.Ю. Методи і протоколи обміну даними сенсорних мереж. Зб. наук. пр. ІПМЕ НАН України. Вип.46. К.,2008. С. 176–183.

35. Березький О.М., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма - Комп’ютерна інженерія". Тернопіль: ЗУНУ, 2024. 32 с.