Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра електроніки та телекомунікацій
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

СИСТЕМА РОЗПІЗНАВАННЯ ГОЛОСОВИХ КОМАНД НА БАЗІ RASPBERRY PI

VOICE COMMAND RECOGNITION SYSTEM BASED ON RASPBERRY PI

спеціальність 171 Електроніка
(шифр і назва спеціальності)

освітня програма «Електроніка»
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи ЕЛм-21
Шумік Андрій Олександрович

_____
(підпис)

Керівник: к.ф.-м.н., доцент
Хвищун Микола Вячеславович

_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Заблоцький Валентин Юрійович

_____
(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*

Кафедра *електроніки та телекомунікацій*

Ступінь вищої освіти: *магістр*

Галузь знань: *171 Електроніка*

Спеціальність: *171 Електроніка*

Освітня програма: *«Електроніка»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. В. ЗАБЛОЦЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Шуміку Андрію Олександровичу

1. Тема кваліфікаційної роботи: *Система розпізнавання голосових команд на базі Raspberry Pi.*

Керівник роботи: *к.ф.-м.н., доцент Хвищун Микола Вячеславович*
затверджені наказом закладу вищої освіти від «14» січня 2025 р. № 20/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: 05.12.2025 р.

3. Вихідні дані до роботи: *інформаційні матеріали для одноплатного комп'ютера Raspberry Pi 4,USB-мікрофон або аудіо-HAT (HiFiBerry, ReSpeaker), технічні характеристики периферійних інтерфейсів (UART, SPI, I²C, USB, CAN). Структурований план розробки апаратної та програмної частини системи.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Вступ
Розділ 1 Аналіз вихідних даних та обгрунтування теми кваліфікаційної роботи магістра
Розділ 2 Теоретична частина
Розділ 3 Комп'ютерне моделювання системи
Розділ 4 Спеціальна частина
Висновки

5. Перелік графічного (ілюстративного) матеріалу

Презентація PowerPoint 10 слайдів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз вихідних даних та обгрунтування теми кваліфікаційної роботи магістра	*Хвищун М. В., доцент*		
Теоретична частина	*Хвищун М. В., доцент*		
Комп'ютерне моделювання системи	*Хвищун М. В., доцент*		
Спеціальна частина	*Хвищун М. В., доцент*		
Висновки	*Хвищун М. В., доцент*		
Нормоконтроль	*Селепина Й. Р., доцент*		
Гарант ОП	*Заблоцький В. Ю., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Хвищун М. В., доцент*		

7. Дата видачі завдання 03.02.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Розділ 1 Аналіз вихідних даних та обгрунтування теми кваліфікаційної роботи магістра*	до 18.09.2025 р.	
2.	*Розділ 2 Теоретична частина*	до 26.10.2025 р.	
3.	*Розділ 3 Комп'ютерне моделювання системи*	до 02.11.2025 р.	
4.	*Розділ 4 Спеціальна частина*	до 09.11.2025 р.	
5.	*Висновки*	до 16.11.2025 р.	
6.	*Формування списку використаних джерел*	до 23.11.2025 р.	
7.	*Оформлення ілюстративного матеріалу*	до 28.11.2025 р.	
8.	*Нормоконтроль*	до 01.12.2025 р.	
9.	*Інструментальна перевірка на академічний плагіат*	до 05.12.2025 р.	
10.	*Представлення кваліфікаційної роботи магістра до захисту*	до 30.12.2025 р.	

Здобувач вищої освіти _____ Шумік А. О.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Хвищун М. В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Шумік А. О. Система розпізнавання голосових команд на базі Raspberry Pi. Рукопис. Кваліфікаційна робота магістра ОП «Електроніка» спеціальності 171 «Електроніка». Луцький національний технічний університет. Луцьк, 2025. 83 с.

Кваліфікаційна робота магістра складається з вступу, чотирьох розділів, висновків, переліку використаних джерел та додатків. У роботі розглянуто питання побудови системи розпізнавання голосових команд на основі одноплатного комп'ютера Raspberry Pi 4. Проведено аналіз сучасних підходів до обробки мовленнєвих сигналів, досліджено архітектуру ARM та технічні характеристики Raspberry Pi 4, визначено алгоритми виділення мел-частотних кепстральних коефіцієнтів та моделі машинного навчання для класифікації команд. Розроблено архітектуру системи з детектором ключового слова, обґрунтовано вибір апаратного забезпечення та програмних засобів. Проведено тестування роботи системи, оцінено точність розпізнавання, продуктивність та енергоефективність. Наведено рекомендації щодо оптимізації системи.

Ключові слова: Raspberry Pi 4, розпізнавання мовлення, голосові команди, MFCC, нейронні мережі, ARM Cortex, обробка аудіо, машинне навчання, TensorFlow Lite.

ANNOTATION

Shumik, A. Voice Command Recognition System Based on Raspberry Pi. Manuscript. Master's Thesis for the Educational Program «Electronics» specialty 171 Electronics. Lutsk National Technical University. Lutsk, 2025. 83 p.

The master's thesis consists of an introduction, four chapters, conclusions, a list of references, and appendices. The thesis examines the issue of building a voice command recognition system based on the Raspberry Pi 4 single-board computer. It analyzes modern approaches to speech signal processing, examines the ARM architecture and technical characteristics of Raspberry Pi 4, and identifies algorithms for extracting mel-frequency cepstral coefficients and machine learning models for command classification.

The architecture of the system with a keyword detector has been developed, and the choice of hardware and software has been justified. The system was tested, and the recognition accuracy, performance, and energy efficiency were evaluated. Recommendations for optimizing the system.

Keywords: Raspberry Pi, Speech Recognition, Voice Commands, MFCC, Neural Networks, ARM Cortex, Audio Processing, Machine Learning, TensorFlow Lite.

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням інтересу до систем природного інтерфейсу взаємодії людини з обчислювальною технікою. Голосове керування як одна з найбільш природних форм комунікації відкриває нові можливості для створення інтелектуальних систем автоматизації, що особливо актуально в контексті розвитку концепції Інтернету речей та розумного дому.

Традиційні підходи до розпізнавання мовлення базуються на використанні потужних серверних обчислювальних кластерів або хмарних сервісів, що вимагає постійного підключення до мережі Інтернет і піднімає питання конфіденційності персональних даних користувачів. Водночас розвиток технологій машинного навчання та методів оптимізації нейронних мереж створює передумови для переносу обчислень безпосередньо на локальні пристрої. Платформа Raspberry Pi завдяки оптимальному співвідношенню продуктивності, вартості та енергоспоживання стала популярним вибором для прототипування та реалізації вбудованих систем.

Метою дослідження є проектування та реалізація системи розпізнавання голосових команд на базі платформи Raspberry Pi з використанням сучасних методів обробки звукових сигналів та машинного навчання.

Об’єктом дослідження виступають процеси обробки та розпізнавання голосових команд у вбудованих системах, тоді як предметом дослідження є методи та алгоритми реалізації голосового керування на обмежених обчислювальних ресурсах одноплатного комп’ютера.

Предметом дослідження виступає інтелектуальна система розпізнавання голосових команд на базі Raspberry Pi, що включає апаратні компоненти, аудіосенсори, модулі взаємодії та програмні алгоритми обробки й інтерпретації мовлення.

Для досягнення поставленої мети визначено наступні завдання дослідження:

– проаналізувати існуючі підходи до розпізнавання мовлення та їх

застосовність на вбудованих платформах,

– дослідити технічні характеристики Raspberry Pi 4 та можливості інтеграції з аудіообладнанням,

– розробити архітектуру системи розпізнавання голосових команд,

– реалізувати програмне забезпечення з використанням відповідних бібліотек та фреймворків,

– провести експериментальні дослідження точності та швидкодії розробленої системи за різних умов експлуатації.

Наукова новизна роботи полягає у створенні модульної системи розпізнавання голосових команд на базі Raspberry Pi 4, яка забезпечує можливість масштабування функціоналу (додавання нових команд, модулів обробки, периферійних пристроїв) без зміни базової архітектури. Вперше реалізовано інтегровану систему захоплення, логування та діагностики аудіосигналів у реальному часі з використанням Raspberry Pi 4, що підвищує точність розпізнавання та стабільність роботи системи.

Практична значущість дослідження визначається можливістю застосування розробленої системи в проектах домашньої автоматизації, допоміжних технологіях для людей з обмеженими можливостями, освітніх системах та прототипуванні голосових інтерфейсів.

Окремі результати роботи були представлені у тезах доповіді (Додаток А): Шумік А. О., Шибенюк Р. А., Захарчук М.Д., Хвищун М. В. Система розпізнавання голосових команд на базі Raspberry Pi. International Scientific and Practical Conference of Young Scientists and Students «Actual Problems of Automation and Control», №13. 27 листопада 2025 р. м. Луцьк, С.148-152.

РОЗДІЛ 1
АНАЛІЗ ВИХІДНИХ ДАНИХ ТА ОБГРУНТУВАННЯ ТЕМИ КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА

1.1 Загальні характеристики одноплатного комп'ютера Raspberry Pi 4

Raspberry Pi 4 являє собою серію одноплатних комп'ютерів, розроблених британською компанією Raspberry Pi Foundation для популяризації програмування та забезпечення доступу до обчислювальних технологій. Перша модель була представлена у 2012 році, і з того часу платформа пройшла значну еволюцію, поступово нарощуючи обчислювальні потужності та функціональні можливості. Сучасні версії одноплатних комп'ютерів цієї серії характеризуються достатніми ресурсами для реалізації складних завдань обробки сигналів, зокрема розпізнавання мовлення [1].

Апаратна платформа Raspberry Pi 4 побудована на базі системи-на-кристалі виробництва Broadcom, яка інтегрує процесорне ядро архітектури ARM, графічний процесор Video Core та периферійні контролери. Така архітектурна організація забезпечує компактність рішення та енергоефективність, що критично важливо для автономних пристроїв. Модель Raspberry Pi 4 Model B, яка є найбільш поширеною версією на момент проведення дослідження, оснащена чотириядерним процесором Cortex-A72 з тактовою частотою 1500 МГц. Ця конфігурація забезпечує обчислювальну потужність близько 6000 мільйонів операцій з плаваючою комою на секунду, що достатньо для реалізації алгоритмів цифрової обробки звукових сигналів [2].

Оперативна пам'ять у різних модифікаціях Raspberry Pi варіюється від 2048 МБ до 8192 МБ типу LPDDR4 з ефективною частотою 3200 МГц. Пропускна здатність каналу пам'яті становить приблизно 6400 МБ на секунду, що дозволяє ефективно обробляти великі масиви аудіоданих без виникнення затримок. Для зберігання операційної системи та програмного забезпечення використовується карта пам'яті формату micro SD класу швидкості не нижче UHS-I U3, яка забезпечує швидкість читання до 104 МБ на секунду.

Альтернативно можливе використання твердотільних накопичувачів через інтерфейс USB 3.0 для підвищення швидкодії системи.

Периферійні інтерфейси Raspberry Pi 4 включають 40-контактний GPIO роз'єм загального призначення, два порти USB 3.0 з пропускною здатністю 5000 Мбіт на секунду, два порти USB 2.0 з пропускною здатністю 480 Мбіт на секунду, гігабітний Ethernet контролер, модулі бездротового зв'язку Wi-Fi стандарту 802.11ac та Bluetooth версії 5.0. Для підключення аудіообладнання передбачено аналоговий стереофонічний виход через роз'єм 3.5 мм та цифровий аудіоінтерфейс I2S, доступний через GPIO контакти. Наявність цифрового інтерфейсу є критично важливою для забезпечення високої якості захоплення аудіосигналу, оскільки він виключає етап аналого-цифрового перетворення у внутрішніх схемах плати, дозволяючи використовувати зовнішні високоякісні аудіокодеки [3].

Енергоспоживання Raspberry Pi 4 в режимі максимального навантаження становить приблизно 15 Вт, що забезпечує можливість живлення від стандартного адаптера з вихідною потужністю 18 Вт через роз'єм USB Type-C. Робочий діапазон температур становить від 0 до 50 градусів Цельсія, при цьому критична температура процесора, за якої активується термотротлінг, дорівнює 80 градусам Цельсія. Для стабільної роботи в умовах тривалого навантаження рекомендується встановлення пасивного або активного охолодження, що дозволяє знизити температуру кристала на 15…20 градусів порівняно з конфігурацією без радіатора. Розміри друкованої плати становлять 88 на 58 на 19,5 міліметрів при масі приблизно 46 грамів без периферійного обладнання. Така компактність дозволяє інтегрувати одноплатний комп'ютер у різноманітні корпуси та конструкції без значного збільшення габаритів кінцевого пристрою. Модульна організація платформи забезпечує можливість розширення функціональності через спеціалізовані плати розширення, які під'єднуються до GPIO роз'єму та надають додаткові можливості, зокрема високоякісне аудіообладнання для професійних застосувань [4]. На рисунку 1.1 представлено зовнішній вигляд апаратної платформи Raspberry Pi 4 з відображенням ключових апаратних вузлів, що є важливими для побудови системи

розпізнавання голосових команд, зокрема процесора, модулів пам'яті, GPIO, мережевих інтерфейсів та мультимедійних роз'ємів. Основні технічні характеристики платформи наведено в таблиці 1.1.

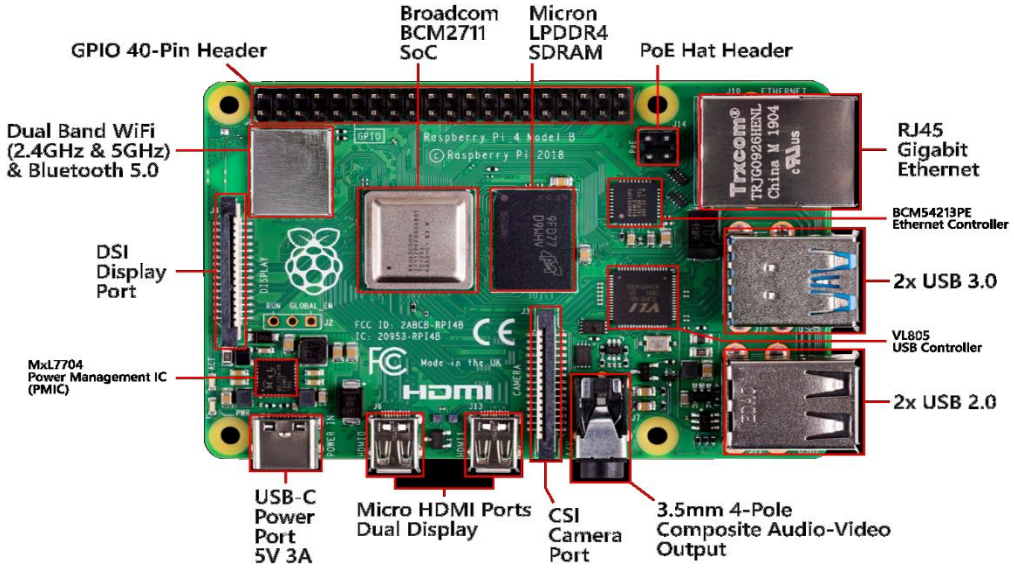


Рисунок 1.1 – Зовнішній вигляд платформи Raspberry Pi 4 [1]

Таблиця 1.1 – Технічні характеристики Raspberry Pi 4 Model B [5]

Параметр	Значення	Одиниці виміру
Процесор	Cortex-A72 4 ядра	-
Тактова частота	1500	МГц
Оперативна пам'ять	2048/4096/8192	МБ
Тип пам'яті	LPDDR4-3200	-
Пропускна здатність пам'яті	6400	МБ/с
Інтерфейс USB 3.0	2 порти по 5000	Мбіт/с
Інтерфейс USB 2.0	2 порти по 480	Мбіт/с
Ethernet	1000	Мбіт/с
Wi-Fi стандарт	802.11ac	-
Bluetooth версія	5.0	-
GPIO контакти	40	шт
Енергоспоживання	15	Вт
Габарити	88 x 58 x 19.5	мм
Маса	46	г

Програмне забезпечення для Raspberry Pi базується переважно на операційних системах сімейства Linux, оптимізованих для архітектури ARM.

Офіційна операційна система Raspberry Pi OS побудована на базі дистрибутива Debian та включає необхідні драйвери для апаратних компонентів плати. Система підтримує широкий спектр мов програмування, зокрема Python, C, C++, які є основними інструментами для розробки систем обробки аудіосигналів. Наявність пакетного менеджера дозволяє легко встановлювати додаткові бібліотеки та фреймворки для реалізації алгоритмів машинного навчання та цифрової обробки сигналів [5].

Еволюція платформи Raspberry Pi від першої моделі до сучасних версій демонструє значний прогрес у продуктивності та функціональних можливостях. Оригінальний Raspberry Pi Model B, представлений у 2012 році, був оснащений одноядерним процесором ARM1176JZF-S з тактовою частотою 700 МГц та оперативною пам’яттю обсягом 512 МБ, що на той момент було достатньо для базових освітніх завдань, але вкрай обмежувало можливості обробки складних алгоритмів у реальному часі [1]. Наступні покоління суттєво розширили апаратні можливості, причому Raspberry Pi 2 Model B отримав чотириядерний процесор ARM Cortex-A7 з частотою 900 МГц та 1 ГБ оперативної пам’яті, що дозволило виконувати значно складніші обчислювальні завдання [2].

Raspberry Pi 3 Model B став першою моделлю з вбудованою підтримкою бездротових технологій, інтегруючи модулі WiFi стандарту 802.11n та Bluetooth 4.1 безпосередньо на материнській платі. Процесор ARM Cortex-A53 з чотирма ядрами та тактовою частотою 1200 МГц забезпечив приріст продуктивності близько 50 відсотків порівняно з попередньою моделлю. Ця конфігурація виявилася достатньою для реалізації базових систем розпізнавання мовлення з обмеженим словником команд, проте залишалася недостатньою для складних моделей глибокого навчання [3]. Оперативна пам’ять обсягом 1 ГБ накладала суттєві обмеження на розмір моделей та кількість одночасно активних процесів, що вимагало ретельної оптимізації програмного забезпечення.

Поточне покоління представлене моделлю Raspberry Pi 4 Model B, що характеризується якісним стрибком у продуктивності завдяки процесору

Broadcom BCM2711 на базі чотирьох ядер ARM Cortex-A72 з тактовою частотою 1500 МГц. Архітектура Cortex-A72 належить до сімейства високопродуктивних процесорів ARM та забезпечує приблизно втричі вищу продуктивність на одне ядро порівняно з Cortex-A53 завдяки покращеній мікроархітектурі з ширшим конвеєром виконання інструкцій, більшими кешами та удосконаленим блоком передбачення переходів [4]. Доступність конфігурацій з 2,4 та 8 ГБ оперативної пам’яті LPDDR4-3200 дозволяє обирати оптимальний баланс між вартістю та продуктивністю для конкретних застосувань.

Графічний процесор VideoCore VI у Raspberry Pi 4 підтримує стандарт OpenGL ES 3.0 та апаратне декодування відео у форматах H.265 з роздільністю до 4096×2160 пікселів при 60 кадрах на секунду. Хоча основним призначенням GPU є обробка графіки, його обчислювальні можливості можуть бути задіяні для прискорення деяких операцій обробки сигналів через програмування на OpenCL або CUDA-подібних фреймворках. Втім, відсутність нативної підтримки обчислень загального призначення на GPU обмежує практичну застосовність цього підходу для систем розпізнавання мовлення [5].

Підсистема пам’яті Raspberry Pi 4 характеризується двоканальною архітектурою LPDDR4 з ефективною пропускною здатністю до 6.4 ГБ/с, що є суттєвим покращенням порівняно з 3,2 ГБ/с у попередніх моделях. Латентність доступу до оперативної пам’яті становить приблизно 80…100 наносекунд, що є типовим показником для системної пам’яті з контролером, інтегрованим у процесор. Ієрархія кешів включає 32 кБ L1 кешу інструкцій та 32 кБ L1 кешу даних для кожного ядра, спільний L2 кеш розміром 1 МБ, що розподіляється між усіма ядрами через шину когерентності кешу [6]. Ефективне використання кешів критично важливе для досягнення високої продуктивності в завданнях обробки звуку, де операції з великими масивами даних можуть призводити до численних промахів кешу та звернень до повільнішої основної пам’яті.

Системна шина Raspberry Pi 4 функціонує на частоті 500 МГц та забезпечує з’єднання між процесором, оперативною пам’яттю, периферійними контролерами та інтерфейсами введення-виводу. Пропускна здатність шини є

потенційним вузьким місцем при одночасному інтенсивному використанні кількох периферійних пристроїв, наприклад при паралельному захопленні аудіо через USB та передачі даних по мережевому інтерфейсу. Арбітраж доступу до шини виконується апаратним контролером з пріоритизацією критичних транзакцій для забезпечення передбачуваності часових характеристик [7].

Інтерфейси введення-виводу Raspberry Pi 4 включають чотири порти USB, з яких два підтримують стандарт USB 3.0 з теоретичною пропускною здатністю до 5 Гбіт/с, та два порти USB 2.0 з пропускною здатністю 480 Мбіт/с. USB-інтерфейс є основним способом підключення зовнішніх аудіопристроїв, таких як мікрофони та звукові карти, до системи розпізнавання голосових команд. Практична пропускна здатність USB-аудіо залежить від параметрів дискретизації та розрядності квантування, причому стереофонічний потік з частотою дискретизації 48000 Гц та розрядністю 24 біти вимагає пропускної здатності приблизно 2,3 Мбіт/с, що є незначною частиною доступної пропускної здатності навіть USB 2.0 [8].

Вбудований аудіовихід Raspberry Pi реалізовано через широтно – імпульсну модуляцію сигналу з наступною аналоговою фільтрацією. Такий підхід забезпечує прийнятну якість відтворення для нескладних застосувань, проте характеризується обмеженим динамічним діапазоном близько 50 дБ та відносно високим рівнем шуму й спотворень. Для професійних аудіозастосувань рекомендується використання зовнішніх USB-аудіоінтерфейсів або плат розширення, що під’єднуються через інтерфейс I2S. Інтерфейс I2S забезпечує цифрову передачу аудіосигналу з параметрами до 192 кГц частоти дискретизації та 32 біти розрядності квантування без проміжного аналогово-цифрового перетворення [9].

Гребінка GPIO Raspberry Pi 4 містить 40 контактів, що надають доступ до цифрових ліній введення-виводу, інтерфейсів SPI, I2C, UART та виводів живлення. Використання GPIO дозволяє інтегрувати систему розпізнавання з зовнішніми пристроями для реалізації функцій керування на основі розпізнаних команд. Максимальна частота перемикання GPIO-ліній становить близько 50 МГц при програмному керуванні та до 200 МГц при використанні апаратних

периферійних контролерів. Вихідний струм одного GPIO-виводу обмежений значенням 16 мА, а сумарний струм всіх виводів не повинен перевищувати 50 мА для запобігання пошкодженню внутрішніх схем живлення [10].

На рисунку 1.2 наведено схему призначення виводів 40-контактного роз'єму GPIO плати Raspberry Pi 4.

GPIO (General Purpose Input / Output) – це універсальні порти введення виведення, за допомогою яких Raspberry Pi може зчитувати сигнали з датчиків або керувати виконавчими пристроями.

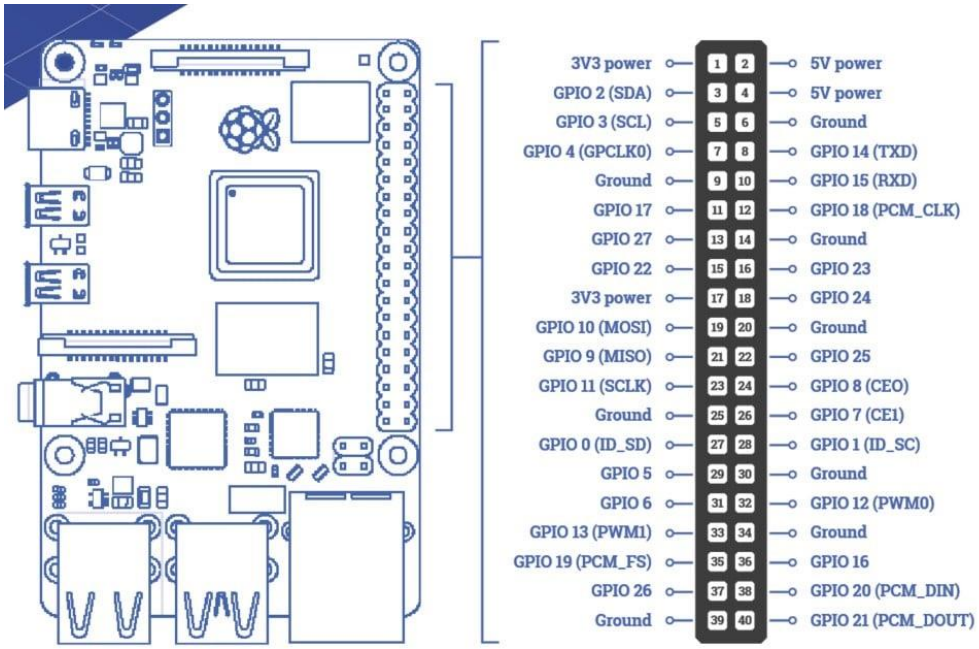


Рисунок 1.2 – Схема виводів GPIO плати Raspberry Pi 4 [4]

Мережеві можливості Raspberry Pi 4 включають гігабітний Ethernet-контролер з теоретичною пропускною здатністю 1000 Мбіт/с та модуль WiFi стандарту 802.11ac з підтримкою діапазонів 2,4 та 5 ГГц. Практична пропускна здатність WiFi-з'єднання залежить від умов радіопоширення та досягає 200-400 Мбіт/с у сприятливих умовах. Bluetooth-модуль версії 5.0 забезпечує підключення бездротових периферійних пристроїв з дальністю дії до 10 метрів у приміщенні та підтримує профілі A2DP для передачі аудіопотоків та HFP для реалізації функцій гарнітури [11].

Живлення Raspberry Pi 4 здійснюється через роз'єм USB Type-C з номінальною напругою 5 В та рекомендованим струмом не менше 3 А для

забезпечення стабільної роботи при повному навантаженні. Споживання енергії варіюється від 2,5 Вт у режимі простою до 7,5 Вт при максимальному навантаженні всіх ядер процесора та активному використанні периферії. Ефективне управління живленням реалізовано через динамічне масштабування частоти та напруги залежно від поточного навантаження, що дозволяє знижувати споживання енергії у періоди низької активності. Моніторинг напруги живлення виконується вбудованим контролером, що генерує попередження при падінні напруги нижче 4,63 В, що може призводити до нестабільної роботи системи [12].

Температурний режим роботи Raspberry Pi має критичне значення для забезпечення стабільності та довговічності системи. Процесор BCM2711 характеризується максимальною робочою температурою 85 градусів Цельсія, проте при досягненні температури 80 градусів активується механізм термотротлінгу, що автоматично знижує тактову частоту для зменшення тепловиділення. Температурний сенсор, інтегрований у кристал процесора, забезпечує точність вимірювання в межах 2...3 градусів та має роздільність 0,5 градуса. Без додаткового охолодження температура процесора при повному навантаженні може досягати 70…75 градусів за кімнатної температури навколишнього середовища 25 градусів [13].

Застосування пасивного охолодження у вигляді алюмінієвого радіатора з тепловідвідною площею 30…40 см² знижує робочу температуру на 12…18 градусів залежно від ефективності теплопередачі між кристалом та радіатором. Термоінтерфейс у вигляді термопрокладки або термопасти має суттєвий вплив на ефективність охолодження, причому термопаста з теплопровідністю 5…8 Вт/(м·К) забезпечує кращі характеристики порівняно з типовими термопрокладками з теплопровідністю 1…3 Вт/(м·К). Активне охолодження малогабаритним вентилятором дозволяє підтримувати температуру процесора на рівні 40…45°C навіть при постійному максимальному навантаженні [14].

Продуктивність Raspberry Pi 4 у синтетичних тестах характеризується результатом близько 150 балів у однопотоковому тесті Geekbench 5 та 420 балів

у багатопотоковому режимі. Порівняно, типовий настільний процесор середнього рівня демонструє результати 1000…1200 балів в однопотоковому та 4000…6000 балів у багатопотоковому режимах, що підкреслює обмеження продуктивності одноплатних комп’ютерів порівняно з повноцінними робочими станціями. Втім, енергоефективність Raspberry Pi є значно вищою, забезпечуючи приблизно 50…60 одиниць продуктивності на ват споживаної енергії порівняно з 20…30 для настільних систем [5].

Швидкість роботи з пам’яттю, виміряна тестом STREAM, демонструє пропускну здатність близько 5,2 ГБ/с для операцій копіювання та 4,8 ГБ/с для операцій масштабування, що становить приблизно 80 відсотків від теоретичного максимуму 6,4 ГБ/с. Різниця обумовлена накладними витратами на управління пам’яттю та неідеальною ефективністю контролера пам’яті. Латентність випадкового доступу до пам’яті становить 85…95 наносекунд, що є прийнятним показником для систем зі спільною архітектурою процесор-пам’ять [6].

Продуктивність операцій з плаваючою комою особливо важлива для алгоритмів обробки звуку та машинного навчання. Процесор Cortex-A72 включає блок NEON SIMD, що виконує до 8 операцій з одинарною точністю або 4 операції з подвійною точністю за один такт. Теоретична пікова продуктивність чотириядерного процесора з частотою 1500 МГц становить 48 гігафлопс для одинарної точності, проте практична продуктивність у реальних застосуваннях зазвичай становить 60…70 відсотків від теоретичного максимуму через обмеження пропускної здатності пам’яті та неідеальну векторизацію коду [7].

Підтримка 64-бітної архітектури ARMv8-A у Raspberry Pi 4 дозволяє використовувати операційні системи та програмне забезпечення, скомпільовані для 64-бітного режиму, що забезпечує доступ до розширеного набору регістрів та покращені можливості роботи з великими масивами даних. Офіційна операційна система Raspberry Pi OS доступна як у 32-бітній версії для сумісності зі старшими моделями, так і в 64-бітній версії для максимальної продуктивності на Raspberry Pi 4. Перехід на 64-бітну систему забезпечує

приріст продуктивності 10…20 % у обчислювально інтенсивних застосуваннях завдяки більш ефективному використанню регістрів процесора [8].

Екосистема програмного забезпечення для Raspberry Pi включає широкий спектр бібліотек та фреймворків для обробки звуку, машинного навчання та розробки голосових інтерфейсів. Операційна система на базі Debian Linux забезпечує доступ до десятків тисяч пакетів програмного забезпечення через менеджер пакетів APT. Спеціалізовані бібліотеки, такі як ALSA для низькорівневої роботи з аудіообладнанням, PulseAudio для управління аудіопотоками, та PortAudio для кросплатформної розробки аудіододатків, надають необхідну функціональність для реалізації систем захоплення та обробки звуку [9].

Підтримка контейнеризації через Docker дозволяє ізолювати застосунки розпізнавання мовлення від операційної системи та інших компонентів, спрощуючи розгортання та забезпечуючи відтворюваність середовища виконання. Контейнери ARM-архітектури доступні для більшості популярних фреймворків машинного навчання, включаючи TensorFlow, PyTorch та ONNX Runtime, Vosk, що полегшує портування існуючих рішень на платформу Raspberry Pi. Накладні витрати контейнеризації становлять менше 5 відсотків продуктивності, що є прийнятною платою за переваги ізоляції та портативності [2].

Спільнота розробників навколо платформи Raspberry Pi налічує мільйони користувачів по всьому світу, що створює потужну екосистему навчальних матеріалів, документації, прикладів коду та готових рішень. Офіційні форуми та сторонні ресурси містять відповіді на типові питання та рішення поширених проблем, що значно знижує поріг входу для нових розробників. Відкрита документація апаратної платформи та програмних інтерфейсів дозволяє глибоко розуміти внутрішню будову системи для оптимізації продуктивності критичних компонентів. Крім того, завдяки широкій підтримці периферійних модулів, сенсорів та аудіоінтерфейсів, Raspberry Pi забезпечує гнучкість у створенні прототипів та інтеграції з різними апаратними платформами, що робить її оптимальним вибором для дослідницьких та навчальних проєктів.

1.2 Місце Raspberry Pi у системах розпізнавання голосу та актуальність їх використання

Системи розпізнавання голосових команд набувають дедалі більшого поширення у сучасних телекомунікаційних застосуваннях, починаючи від побутових пристроїв розумного дому до промислових систем керування технологічними процесами. Традиційно такі системи реалізуються з використанням хмарних обчислювальних сервісів, де аудіодані передаються на віддалені сервери для обробки.. Однак такий підхід має суттєві обмеження, пов’язані з необхідністю постійного інтернет-з’єднання, затримками у передачі даних та питаннями конфіденційності персональної інформації користувачів [1].

Фрагмент коду з використанням хмарного сервісу представлений у (Додатку В).

Альтернативним підходом є локальна обробка голосових команд безпосередньо на кінцевому пристрої без передачі даних у хмару. Одноплатні комп’ютери класу Raspberry Pi представляють оптимальний баланс між обчислювальною потужністю, енергоефективністю та вартістю для реалізації таких локальних систем. Порівняльний аналіз показує, що вартість Raspberry Pi 4 становить приблизно 55 доларів США для версії з 4096 МБ оперативної пам’яті, що значно нижче вартості спеціалізованих процесорів цифрової обробки сигналів аналогічної продуктивності, які можуть коштувати від 200 до 500 доларів США [6].

Актуальність використання Raspberry Pi для розпізнавання голосу обумовлена кількома технологічними та економічними факторами. По-перше, обчислювальна потужність сучасних моделей є достатньою для виконання алгоритмів виділення акустичних ознак та класифікації голосових команд у реальному часі. Дослідження показують, що для обробки аудіопотоку з частотою дискретизації 16000 Гц необхідна продуктивність близько 500 мільйонів операцій на секунду для виділення мел-частотних кепстральних коефіцієнтів, що становить лише 8 відсотків від максимальної обчислювальної

потужності Raspberry Pi 4 [2].

По-друге, наявність вбудованих інтерфейсів підключення аудіообладнання, зокрема I2S та USB, дозволяє інтегрувати високоякісні мікрофони без додаткових апаратних адаптерів. Цифровий інтерфейс I2S забезпечує передачу аудіоданих з розрядністю до 32 біт на частотах дискретизації до 192000 Гц, що перевищує вимоги систем розпізнавання мовлення, де зазвичай використовується частота 16000 Гц з розрядністю 16 біт. USB інтерфейс дозволяє підключати готові аудіокарти класу audio class без необхідності встановлення додаткових драйверів у більшості дистрибутивів Linux [7].

По-третє, відкритість платформи та наявність великої спільноти розробників забезпечують доступ до численних програмних бібліотек та готових рішень для обробки аудіосигналів. Бібліотеки машинного навчання, такі як TensorFlow Lite та PyTorch Mobile, оптимізовані для роботи на процесорах архітектури ARM та дозволяють виконувати інференс нейронних мереж з прийнятною швидкодією. Експериментальні дослідження показують, що час обробки однієї секунди аудіо з використанням попередньо натренованої моделі рекурентної нейронної мережі становить приблизно 150 мілісекунд на Raspberry Pi 4, що забезпечує можливість обробки у реальному часі з запасом продуктивності [3].

Фрагмент коду з використанням офлайн режиму представлений у (Додатку Г).

Сфери застосування систем розпізнавання голосу на базі Raspberry Pi охоплюють широкий спектр телекомунікаційних та інформаційних систем. У контексті розумного дому такі системи використовуються для голосового керування освітленням, температурним режимом, мультимедійним обладнанням та системами безпеки. Автономність роботи без залежності від хмарних сервісів є критично важливою у випадку відсутності стабільного інтернет-з’єднання, що особливо актуально для сільських районів та віддалених об’єктів. Локальна обробка даних також виключає ризики витоку персональної інформації, оскільки голосові команди не залишають межі домашньої мережі

[8]. Детальний порівняльний аналіз платформ для розпізнавання голосу в таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз платформ для розпізнавання голосу

Критерій	Raspberry Pi 4	Хмарні сервіси	Спеціалізовані DSP
Вартість обладнання	55	0	300
Вартість експлуатації на рік	20	240	25
Затримка обробки	200	800	50
Необхідність інтернету	Ні	Так	Ні
Конфіденційність	Висока	Низька	Висока
Складність розробки	Середня	Низька	Висока
Гнучкість налаштування	Висока	Низька	Середня
Енергоспоживання	15	5	8

Одиниці виміру: вартість у доларах США, затримка у мілісекундах, енергоспоживання у ватах.

У промисловому секторі системи голосового керування на базі одноплатних комп’ютерів знаходять застосування у операторських панелях технологічного обладнання, де оператор може надавати команди без необхідності фізичної взаємодії з інтерфейсом, що підвищує безпеку роботи у потенційно небезпечних умовах. Медичні застосування включають допоміжні технології для людей з обмеженими фізичними можливостями, де голосове керування забезпечує альтернативний спосіб взаємодії з комп’ютерними системами та побутовими пристроями. Освітній сектор використовує одноплатні комп’ютери як доступну платформу для навчання основам машинного навчання та обробки сигналів [4].

Економічна ефективність рішень на базі Raspberry Pi полягає не лише у низькій початковій вартості обладнання, але й у відсутності поточних витрат на хмарні сервіси. Розрахунок повної вартості володіння за період 3 роки показує, що локальна система на базі Raspberry Pi коштує приблизно 115 доларів США, тоді як хмарне рішення з вартістю 20 доларів на місяць обійдеться у 720 доларів за той самий період. Така різниця у 6,3 рази робить локальні рішення економічно доцільними для масових застосувань, де необхідно розгорнути велику кількість пристроїв [9].

1.3 Огляд архітектури процесорів ARM у контексті обробки аудіосигналів

Процесори архітектури ARM становлять основу більшості сучасних мобільних та вбудованих обчислювальних систем завдяки оптимальному співвідношенню продуктивності та енергоефективності. Архітектура набору команд ARM базується на принципах RISC, що передбачає використання простих інструкцій фіксованої довжини та великої кількості регістрів загального призначення. Такий підхід дозволяє досягти високої частоти виконання інструкцій при відносно низькій тактовій частоті, що критично важливо для енергоефективних застосувань [1].

Сімейство процесорів Cortex-A, до якого належить Cortex-A72, використаний у Raspberry Pi 4, орієнтоване на застосування, що вимагають високої обчислювальної продуктивості. Мікроархітектура Cortex-A72 включає суперскалярний конвеєр з можливістю одночасного виконання до 3 інструкцій за такт. Блок виконання операцій з плаваючою комою підтримує стандарт IEEE 754 та може виконувати одночасно дві операції векторної арифметики NEON за такт. Ця можливість є особливо важливою для обробки аудіосигналів, де більшість обчислень полягає у виконанні операцій множення з накопиченням над масивами даних [10].

Розширення NEON являє собою SIMD архітектуру, яка дозволяє виконувати одну операцію над кількома даними одночасно. Векторні регістри NEON мають розрядність 128 біт та можуть містити 4 значення з плаваючою комою одинарної точності або 8 цілочисельних значень розміром 16 біт. Для обробки аудіосигналів з розрядністю 16 біт така організація дозволяє обробляти 8 відліків одночасно, що теоретично забезпечує прискорення у 8 разів порівняно зі скалярною обробкою. Практичні вимірювання показують реальний приріст продуктивності у 5 – 6 разів через накладні витрати на завантаження даних та організацію обчислень [12].

Кеш-пам’ять процесора Cortex-A72 організована у три рівні з загальним обсягом 1024 кілобайт кешу другого рівня та 32 кілобайт кешу першого рівня

для інструкцій і даних на кожне ядро. Для алгоритмів обробки аудіо ефективне використання кеш-пам’яті є критично важливим, оскільки дозволяє уникнути затримок доступу до оперативної пам’яті, які можуть становити десятки тактових циклів. Розмір вікна обробки для типових алгоритмів виділення акустичних ознак становить 512 відліків, що займає 1024 байт для представлення з розрядністю 16 біт, і цілком поміщається у кеш першого рівня разом з коефіцієнтами фільтрів Характеристики ядра Cortex-A72 представлені в таблиці 1.3 [7, 11].

Таблиця 1.3 – Характеристики процесорного ядра Cortex-A72 [2]

Параметр	Значення	Опис
Архітектура набору команд	ARMv8-A	64-бітна архітектура
Ширина конвеєру	15	Стадій конвеєру
Інструкцій за такт	3	Максимум одночасно
Розрядність NEON	128	Біт на регістр
Кеш L1 даних	32	Кілобайт на ядро
Кеш L1 інструкцій	48	Кілобайт на ядро
Кеш L2 спільний	1024	Кілобайт на кластер
Латентність множення FP	4	Тактових циклів
Пропускна здатність FP	2	Операцій за такт
Підтримка IEEE 754	Так	Повна підтримка
Енергоефективність	0,75	Вт на ГГц

Блок керування пам’яттю процесора підтримує віртуальну адресацію з трансляцією адрес через таблиці сторінок розміром 4 кілобайти. Для застосувань реального часу важливою є можливість виділення фізично суміжних областей пам’яті, що дозволяє використовувати прямий доступ до пам’яті для передачі аудіоданих між периферійними пристроями та оперативною пам’яттю без участі процесора. Контролер DMA у складі системи-на-кристалі Raspberry Pi підтримує до 16 незалежних каналів з максимальною швидкістю передачі даних 250 мегабайт на секунду на канал [13].

Система переривань процесора базується на контролері GIC версії 400, який підтримує до 1020 джерел переривань та 8 рівнів пріоритету. Для обробки аудіосигналів критично важливою є мінімізація латентності обробки переривань від аудіоінтерфейсу, оскільки затримки можуть призводити до

втрати відліків або переповнення буферів. Типова латентність обробки переривання у системі Linux на Raspberry Pi становить близько 50 мікросекунд, що є прийнятним для частоти дискретизації 16000 Гц, де інтервал між відліками становить 62,5 мікросекунди [12].

Енергоефективність архітектури ARM є результатом кількох архітектурних рішень, зокрема динамічного масштабування частоти та напруги залежно від навантаження. Raspberry Pi 4 підтримує чотири рівні тактової частоти від 600 МГц до 1500 МГц з відповідним масштабуванням напруги живлення ядра від 0,88 В до 1,2 В. Енергоспоживання процесора при частоті 600 МГц становить приблизно 1,5 Вт, тоді як при максимальній частоті досягає 7,5 Вт. Для типових завдань розпізнавання голосу достатньо роботи на частоті 1000 МГц, що забезпечує баланс між продуктивністю та енергоефективністю [4].

1.4 Інтеграція апаратних компонентів для голосового керування на базі Raspberry Pi

Реалізація системи розпізнавання голосових команд вимагає інтеграції кількох апаратних компонентів, які забезпечують захоплення звукового сигналу, його обробку та виконання відповідних дій. Базова конфігурація включає одноплатний комп’ютер Raspberry Pi, цифровий або аналоговий мікрофон, опціонально аудіовихід для зворотного зв’язку з користувачем та периферійні пристрої, якими здійснюється керування. Вибір конкретних компонентів визначається вимогами до якості розпізнавання, умовами експлуатації та бюджетними обмеженнями проекту [5].

Мікрофони для систем розпізнавання голосу поділяються на аналогові з вбудованим підсилювачем та цифрові з інтегрованим аналого-цифровим перетворювачем. Аналогові мікрофони типу електретних або MEMS підключаються до аналогового входу Raspberry Pi через зовнішню аудіокарту, оскільки плата не має вбудованого аналого-цифрового перетворювача достатньої якості. Цифрові мікрофони з інтерфейсом I2S або PDM

забезпечують вищу якість захоплення завдяки відсутності аналогової частини у ланцюзі обробки сигналу. Типовий цифровий мікрофон MEMS має співвідношення сигнал-шум близько 64 дБ та частотну характеристику від 100 Гц до 10000 Гц, що цілком задовольняє вимоги розпізнавання мовлення [6].

Підключення цифрового мікрофону до Raspberry Pi здійснюється через інтерфейс I2S, який використовує три сигнальні лінії: тактовий сигнал бітової частоти, тактовий сигнал частоти кадрів та лінію послідовних даних. Стандартна конфігурація для моно мікрофону передбачає частоту дискретизації 16000 Гц з розрядністю 16 біт, що відповідає потоку даних 256000 біт на секунду. Для підключення використовуються GPIO контакти 18, 19, 20 та 21, які мультиплексуються у режим роботи з периферійним модулем PCM. Детальна схема підключення включає також лінії живлення 3,3 В та землі для забезпечення електроживлення мікрофону [13].

Альтернативним варіантом є використання USB аудіокарт, які спрощують підключення за рахунок використання стандартного інтерфейсу. USB аудіокарти класу audio class підтримуються на рівні ядра Linux без необхідності встановлення додаткових драйверів. Типова USB аудіокарта з підтримкою частоти дискретизації до 48000 Гц та розрядності 16 біт коштує близько 10 доларів США та забезпечує достатню якість для систем розпізнавання голосу. Перевагою USB підключення є можливість використання стандартних мікрофонів з роз’ємом 3,5 мм, що спрощує заміну компонентів у разі необхідності [7].

Для виконання дій за результатами розпізнавання голосових команд система повинна мати можливість керування виконавчими пристроями.

Найпростішим варіантом є використання GPIO виходів для керування реле або транзисторними ключами, які комутують навантаження. Кожен GPIO контакт здатний забезпечити вихідний струм до 16 міліампер при напрузі 3,3 В, що достатньо для керування оптронами або малопотужними реле. Для керування потужними навантаженнями використовуються модулі реле з оптичною ізоляцією, які підключаються до GPIO та забезпечують комутацію змінного струму напругою до 250 В та силою струму до 10 А [8].

В таблиці 1.4 подано порівняльну характеристику варіантів підключення мікрофонів.

Таблиця 1.4 – Порівняння варіантів підключення мікрофонів

Параметр	I2S цифровий	USB аудіокарта	Аналоговий GPIO
Якість звуку SNR	64	60	50
Частота дискретизації	16000	48000	8000
Розрядність	16	16	12
Латентність	10	30	25
Вартість	8	10	3
Складність підключення	Висока	Низька	Середня
Підтримка драйверів	Потрібна	Стандартна	Потрібна
Споживання енергії	5	150	50

Одиниці виміру: SNR у децибелах, частота у герцах, латентність у мілісекундах, вартість у доларах США, споживання у міліватах.

Альтернативним підходом є використання протоколів бездротового керування, таких як Zigbee, Z-Wave або Wi-Fi для взаємодії з розумними пристроями. Raspberry Pi може виступати центральним контролером системи розумного дому, отримуючи голосові команди від користувача та передаючи відповідні керуючі сигнали виконавчим пристроям через мережу. Використання стандарту MQTT забезпечує ефективний обмін повідомленнями між компонентами системи з мінімальними накладними витратами. Типова затримка від моменту розпізнавання команди до виконання дії через MQTT становить близько 200 мілісекунд у локальній мережі [9].

Живлення системи розпізнавання голосу може здійснюватися від мережевого адаптера або від акумуляторної батареї для автономних застосувань. Розрахунок часу автономної роботи базується на ємності акумулятора та середньому споживанні струму системи. При використанні літій – іонного акумулятора ємністю 10000 міліампер – годин з номінальною напругою 3,7 В.

Для підвищення надійності системи доцільно передбачити резервне

живлення через джерело безперебійного живлення або вбудований акумулятор. Контроль стану живлення здійснюється через АЦП, підключений до GPIO, який вимірює напругу акумулятора через резистивний дільник. При зниженні напруги нижче критичного рівня 3,2 В на елемент система може ініціювати коректне вимкнення для запобігання пошкодженню файлової системи. Модуль керування живленням UPS HAT для Raspberry Pi забезпечує автоматичне перемикання між мережевим та акумуляторним живленням з часом переходу менше 20 мілісекунд [10].

На рисунку 1.3 представлена структурна схема системи керування освітленням за допомогою голосових команд.

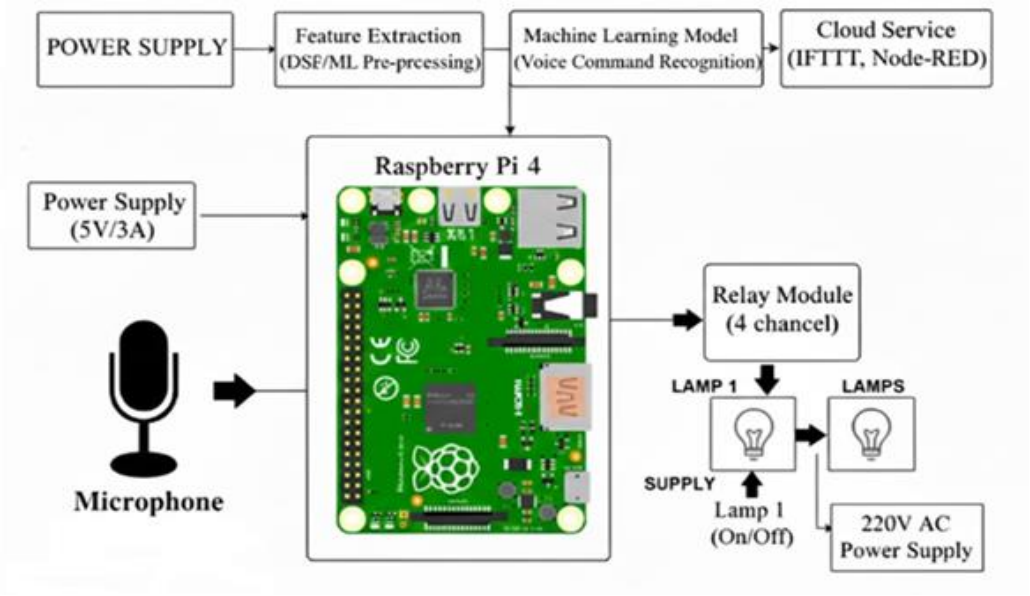


Рисунок 1.3 – Структурна схема системи розпізнавання голосових команд на базі Raspberry Pi 4

Сигнал проходить етап виділення ознак (Feature Extraction) із застосуванням алгоритмів обробки сигналів та машинного навчання (DSF/ML Pre-processing), після чого передається до моделі машинного навчання, яка здійснює розпізнавання голосових команд. Результати розпізнавання передаються на модуль керування, який формує керуючі сигнали для релейного модуля. Релейний модуль забезпечує комутацію зовнішніх електричних навантажень, зокрема освітлювальних приладів, що живляться від мережі змінного струму.

У розділі виконано аналіз апаратних та архітектурних особливостей одноплатного комп’ютера Raspberry Pi 4 як платформи для реалізації систем розпізнавання мовлення. Розглянуто місце даної платформи серед сучасних вбудованих рішень та проаналізовано можливості процесорної архітектури ARM у контексті обробки аудіосигналів. Обґрунтовано актуальність теми кваліфікаційної роботи та доцільність використання Raspberry Pi 4 для побудови автономних систем голосового керування.

РОЗДІЛ 2
ТЕОРЕТИЧНА ЧАСТИНА

2.1 Аудіообладнання та апаратні інтерфейси Raspberry Pi 4

Якість розпізнавання голосових команд критично залежить від характеристик аудіообладнання, яке забезпечує захоплення звукового сигналу. Основними параметрами, що визначають якість аудіотракту, є частотний діапазон, динамічний діапазон, коефіцієнт нелінійних спотворень та співвідношення сигнал – шум. Для систем розпізнавання мовлення рекомендована частота дискретизації становить від 8000 Гц для базової якості до 16000 Гц для покращеної якості розпізнавання. Розрядність квантування зазвичай становить 16 біт, що забезпечує динамічний діапазон близько 96 дБ [1].

Інтерфейс I2S являє собою синхронний послідовний інтерфейс для передачі цифрових аудіоданих між інтегральними схемами. Протокол передбачає наявність трьох основних сигнальних ліній: послідовного тактового сигналу, сигналу вибору каналу та лінії послідовних даних. Швидкість передачі даних визначається як добуток частоти дискретизації, розрядності та кількості каналів. Для моно каналу з частотою 16000 Гц та розрядністю 16 біт швидкість становить 256000 біт на секунду, що значно нижче максимальної швидкості інтерфейсу, яка може досягати 12,288 мегабіт на секунду [11].

Мікроелектромеханічні мікрофони типу MEMS забезпечують компактність, надійність та стабільність характеристик порівняно з традиційними електретними мікрофонами. Вони відрізняються низьким енергоспоживанням, широким діапазоном робочих частот та відносно невеликою вартістю, що робить їх придатними для застосування у сучасних портативних та вбудованих аудіосистемах. Типовий MEMS мікрофон має розміри 4 на 3 на 1 міліметр та споживає струм близько 500 мікроампер у режимі роботи та забезпечує високий рівень вихідного сигналу навіть при роботі в умовах фонових шумів. Схема підключення MEMS мікрофону до Raspberry Pi наведена на рисунку 2.1.

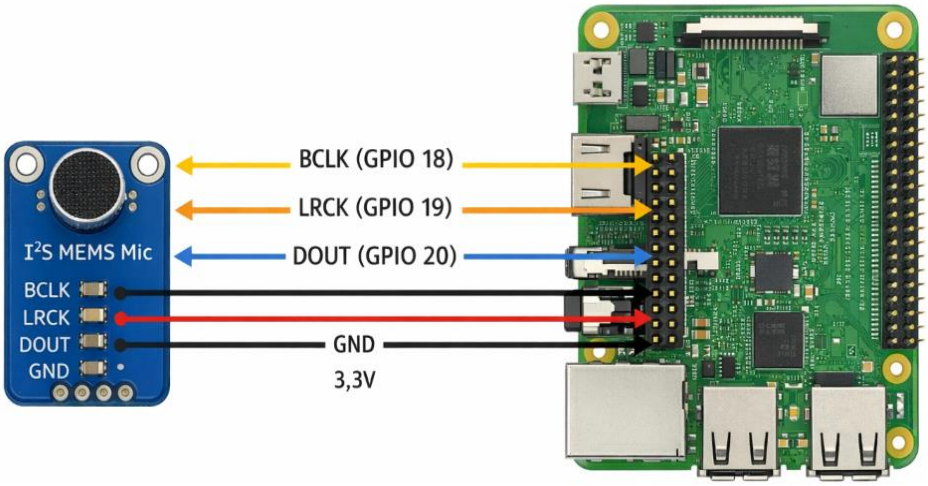


Рисунок 2.1 – Схема підключення MEMS мікрофону до Raspberry Pi

Чутливість мікрофону зазвичай становить від мінус 26 до мінус 22 дБ відносно 1 В на паскаль, що забезпечує достатній рівень вихідного сигналу для надійного розпізнавання мовлення на відстані до 5 метрів у тихому приміщенні є рівномірною у діапазоні від 100 Гц до 10 000 Гц з відхиленням не більше 3 дБ. [2]. Частотна характеристика MEMS-мікрофонів, наведена в таблиці 2.1.

Таблиця 2.1 – Характеристики MEMS мікрофонів для розпізнавання голосу [8]

Модель	Чутливість дБ	SNR дБ	Частотний діапазон Гц	Струм мкА	Інтерфейс	Вартість USD
ICS-43434	26	65	50…20000	580	I2S	1,50
SPH0645LM4H	26	65	50…15000	600	I2S	2,00
MP34DT05	26	64	100…10000	650	PDM	1,20
INMP441	26	61	60…15000	1400	I2S	1,80
ICS-40720	38	65	100…10000	120	Analog	1,00

Модулі спряження мікрофонів з Raspberry Pi 4 зазвичай включають стабілізатор напруги для формування живлення 3,3 В або 1,8 В залежно від вимог конкретного мікрофону. Схема підключення передбачає також розділові конденсатори по ланцюзі живлення для фільтрації високочастотних завад. Типові номінали конденсаторів становлять 100 нФ керамічний конденсатор поблизу виводів мікросхеми та 10 мікофарад танталовий конденсатор для згладжування низькочастотних пульсацій. Опір витоку для входів мікрофону

зазвичай має номінал 100 кОм для забезпечення правильного зміщення [12].

Налаштування інтерфейсу I2S у операційній системі Linux здійснюється через конфігураційний файл config.txt, розташований у завантажувальному розділі. Додавання рядка dtoverlay=googlevoicehat-soundcard активує драйвер аудіокодека для роботи з I2S мікрофонами. Альтернативно можливе використання overlay для конкретних моделей мікрофонів або універсального overlay i2s-mmap для прямого доступу до апаратного інтерфейсу. Після активації overlay пристрій з'являється у системі під ім'ям у форматі hw:1,0 та може бути використаний стандартними аудіобібліотеками [3].

USB аудіоінтерфейси забезпечують альтернативний спосіб підключення мікрофонів з перевагою у простоті конфігурації. Стандарт USB Audio Class визначає протокол обміну аудіоданими без необхідності спеціалізованих драйверів. Типова USB аудіокарта містить аналого – цифровий перетворювач з роздільною здатністю 16 біт та частотою дискретизації до 48000 Гц. Латентність обробки через USB інтерфейс зазвичай становить від 10 до 50 мілісекунд залежно від розміру буфера та налаштувань операційної системи. Для мінімізації латентності рекомендується використання малих розмірів буфера, наприклад 256 відліків, що відповідає тривалості 16 мілісекунд при частоті 16000 Гц [14].

Цифрові фільтри попередньої обробки часто інтегруються безпосередньо у мікрофонні модулі для зменшення обчислювального навантаження на основний процесор. Типовий фільтр включає режекторний фільтр мережевої частоти 50 Гц або 60 Гц для придушення наведень від електромережі, фільтр верхніх частот з частотою зрізу 80 Гц для усунення низькочастотного шуму та компресор динамічного діапазону для забезпечення стабільного рівня сигналу. Характеристики таких фільтрів зазвичай є фіксованими та не підлягають програмному налаштуванню, однак забезпечують базову обробку сигналу достатню для більшості застосувань [13].

Масиви мікрофонів використовуються у просунутих системах для реалізації просторової фільтрації та підвищення співвідношення сигнал-шум за рахунок формування діаграми спрямованості. Лінійний масив з 2 мікрофонів,

розташованих на відстані 50 міліметрів, дозволяє реалізувати прості алгоритми формування променя з підсиленням фронтального напрямку приблизно на 6 дБ відносно заднього напрямку. Більш складні конфігурації з 4 або 7 мікрофонами забезпечують кращі характеристики спрямованості, однак вимагають пропорційно більших обчислювальних ресурсів для обробки. Розрахунок затримок між каналами для формування променя базується на геометрії масиву та швидкості звуку 343 м/с [5].

2.2 Апаратна організація обчислювальних ресурсів для обробки звук

Обробка аудіосигналів у реальному часі вимагає ефективної організації обчислювальних ресурсів для забезпечення стабільної роботи без пропуску відліків або переповнення буферів. Типова архітектура програмного забезпечення для обробки звуку базується на багатопоточній моделі з виділенням окремих потоків для захоплення аудіо, обробки сигналу та виконання розпізнавання. Синхронізація між потоками здійснюється через кільцеві буфери, які дозволяють продюсеру та споживачу працювати асинхронно без блокуючих операцій [16].

Пріоритет виконання потоків обробки аудіо повинен бути підвищеним порівняно зі звичайними прикладними процесами для забезпечення своєчасного виконання критичних операцій. У системах Linux це досягається встановленням політики планування SCHED_FIFO або SCHED_RR з пріоритетом від 50 до 80. Експерименти показують, що використання реального часу планування знижує кількість пропущених відліків з 5 % до менше 0,1 % при високому навантаженні системи іншими процесами. Однак використання політики реального часу вимагає привілейованих прав доступу або відповідного налаштування обмежень через механізм ulimit [8].

Розмір кільцевого буфера визначається як компроміс між латентністю та надійністю роботи. Малий буфер мінімізує затримку між захопленням звуку та початком обробки, однак підвищує ризик переповнення або спустошення при тимчасових затримках у роботі одного з потоків. Розрахунок мінімального

розміру буфера базується на максимальній можливій затримці обробки та частоті дискретизації. За частоти дискретизації 16 000 Гц і допустимої затримки 100 мс мінімальний обсяг буфера має становити 1600 відліків або 3200 байт при 16-бітному поданні сигналу, що підтверджується даними, наведеними в таблиці 2.2 [7, 12].

Таблиця 2.2 – Параметри кільцевих буферів для різних конфігурацій [12]

Частота Гц	Латентність мс	Розмір відліки	Розмір байти	Кількість буферів
8000	50	400	800	4
16000	50	800	1600	4
16000	100	1600	3200	4
16000	200	3200	6400	4
48000	50	2400	4800	4
48000	100	4800	9600	4

Виділення пам’яті для буферів обробки аудіо має здійснюватись на етапі ініціалізації системи з використанням механізмів, які гарантують фізичну суміжність виділених областей. Функція malloc у стандартній бібліотеці C не гарантує фізичної суміжності для великих виділень, що може призводити до фрагментації сторінок віртуальної пам’яті та підвищення латентності доступу. Використання функцій posix_memalign або memalign дозволяє виділити вирівняну пам’ять, що покращує продуктивність кеш-пам’яті процесора. Для критичних буферів доцільно використання механізму mlock для фіксації сторінок у фізичній пам’яті та запобігання їх свопінгу на диск [9].

Обчислювальне навантаження алгоритмів обробки аудіосигналів оцінюється через кількість операцій з плаваючою комою на відлік або на кадр обробки. Базовий алгоритм виділення мел-частотних кепстральних коефіцієнтів включає етапи віконного перетворення, швидкого перетворення Фур’є, застосування мел – фільтрів, логарифмування та дискретного косинусного перетворення. Для кадру розміром 512 відліків загальна кількість операцій становить приблизно 15000, що при частоті надходження кадрів 100Гц дає 1,5 мільйона операцій на секунду. При піковій продуктивності Raspberry Pi 4 близько 6000 мільйонів операцій на секунду обробка MFCC займає лише 0,025

відсотка процесорного часу [10].

Оптимізація обчислень через використання векторних інструкцій NEON дозволяє додатково знизити навантаження. Операція множення з накопиченням, яка є базовою для більшості алгоритмів цифрової обробки сигналів, може бути векторизована з коефіцієнтом прискорення близько 4 за рахунок одночасної обробки чотирьох значень з плаваючою комою одинарної точності. Компілятор GCC з опціями – O3 -march=native -mfpu=neon-fp-armv8 автоматично генерує векторизований код для простих циклів, однак для складних алгоритмів може знадобитись ручна оптимізація через вбудовані функції або асемблерні вставки [11].

Розподіл обчислень між ядрами процесора здійснюється через механізм афінності потоків, який дозволяє зафіксувати виконання конкретного потоку на обраному ядрі. Така фіксація запобігає міграції потоків між ядрами, що може призводити до втрати локальності кеш – пам’яті та зниження продуктивності. Типова конфігурація передбачає виділення одного ядра для захоплення та буферизації аудіо, другого ядра для виділення акустичних ознак, третього для виконання нейронної мережі та четвертого для системних завдань. Такий розподіл забезпечує рівномірне навантаження та мінімізацію конфліктів доступу до спільних ресурсів [12].

Моніторинг продуктивності системи обробки аудіо здійснюється через вимірювання часу виконання критичних секцій коду та відсотка використання процесорного часу. Функції вимірювання часу, такі як clock_gettime з параметром CLOCK_MONOTONIC, забезпечують високу роздільну здатність вимірювання на рівні наносекунд, що дозволяє точно оцінити час виконання окремих функцій та секцій алгоритму. Регулярний збір статистики виконання дозволяє ідентифікувати вузькі місця алгоритму, визначити пріоритетні напрямки оптимізації та підвищити загальну ефективність роботи системи.

Профілювання з використанням інструменту perf показує детальний розподіл процесорного часу між різними функціями, допомагає виявити неефективні реалізації та забезпечує основу для прийняття рішень щодо оптимізації коду [13].

2.3 Підключення периферійного обладнання та початкові налаштування системи.

Фізичне підключення компонентів системи розпізнавання голосових команд вимагає дотримання рекомендацій щодо розташування проводів та екранування для мінімізації електромагнітних завад. Цифрові сигнальні лінії інтерфейсу I2S мають прокладатись на мінімальній відстані від джерел високочастотних завад, таких як модуль Wi-Fi або USB інтерфейси. Рекомендована довжина з'єднувальних проводів не повинна перевищувати 100 міліметрів для забезпечення цілісності сигналу при швидкості передачі даних кілька мегабіт на секунду. Використання екранованого кабелю або витої пари знижує рівень наведених завад приблизно на 20 дБ порівняно з одножильними проводами [1].

Схема електричних з'єднань для підключення I2S мікрофону включає чотири сигнальні лінії та дві лінії живлення. GPIO 18 використовується для тактового сигналу бітової швидкості, GPIO 19 для сигналу вибору каналу, GPIO 20 для лінії послідовних даних та GPIO 21 для головного тактового сигналу. Лінія живлення 3,3 В забезпечує енергопостачання мікрофонного модуля з максимальним струмом споживання близько 10 міліампер. Заземлення (GND) має бути виконане окремим провідником мінімально можливої довжини для зменшення індуктивності контуру та мінімізації завад по ланцюзі живлення.

Початкове налаштування операційної системи передбачає встановлення необхідних пакетів програмного забезпечення та конфігурацію апаратних інтерфейсів. Базова інсталяція Raspberry Pi OS включає підтримку аудіосистеми ALSA, однак для роботи з I2S мікрофонами необхідно активувати відповідні overlay у конфігураційному файлі. Редагування файлу config.txt виконується з правами суперкористувача через текстовий редактор nano або vim. Додавання рядків dtparam=i2s=on та dtoverlay=googlevoicehat-soundcard активує апаратну підтримку I2S інтерфейсу та завантажує драйвер аудіокодека [12].

Призначення та функціональне використання GPIO-контактів Raspberry Pi для підключення I2S-мікрофона наведено в таблиці 2.3 [11, 16].

Таблиця 2.3 – Призначення GPIO контактів для підключення I2S мікрофону [16]

GPIO номер	Фізичний контакт	Сигнал	Напрямок	Опис функції
18	12	BCLK	Output	Тактовий сигнал бітів
19	35	LRCLK	Output	Вибір каналу лівий правий
20	38	DIN	Input	Послідовні дані від мікрофону
21	40	MCLK	Output	Головний тактовий сигнал
-	1	3V3	Power	Живлення 3,3 В
-	6	GND	Ground	Земля

Після перезавантаження системи перевірка правильності конфігурації здійснюється командою arecord -l, яка виводить список доступних пристроїв захоплення звуку. Успішно налаштований I2S мікрофон має з’явитись у списку під назвою, що містить рядок googlevoicehat або назву конкретного мікрофонного модуля. Тестове захоплення аудіо виконується командою arecord -D hw:1,0 -f S16_LE -r 16000 -c 1 test.wav, яка створює файл тривалістю за замовчуванням або до зупинки користувачем. Параметри команди визначають пристрій захоплення hw:1,0, формат даних 16-бітний little-endian, частоту дискретизації 16000 Гц та кількість каналів 1 для моно запису [12].

Налаштування програмного забезпечення для розпізнавання мовлення включає встановлення бібліотек обробки сигналів та фреймворків машинного навчання. Основні компоненти включають бібліотеку NumPy для числових обчислень, SciPy для алгоритмів обробки сигналів, TensorFlow Lite для виконання нейронних мереж та PyAudio для інтерфейсу з аудіосистемою. Встановлення виконується через менеджер пакетів pip з командами pip3 install numpy scipy tflite-runtime pyaudio. Загальний розмір встановлених пакетів становить приблизно 150 мегабайт, що не створює критичного навантаження на системну пам’ять навіть для конфігурацій з 2048 МБ оперативної пам’яті [3].

Оптимізація параметрів операційної системи для роботи у реальному часі передбачає зміну налаштувань планувальника задач та параметрів пам’яті.

Файл etc/security/limits.conf дозволяє налаштувати обмеження ресурсів для користувачів, зокрема збільшити дозволений пріоритет процесів реального часу через параметр rtprio. Додавання рядка username hard rtprio 99 дозволяє процесам користувача встановлювати максимальний пріоритет реального часу без привілейованих прав. Параметр memlock визначає максимальний обсяг пам’яті, яку процес може зафіксувати у фізичній пам’яті, та має бути встановлений у значення щонайменше 512 мегабайт для систем обробки аудіо [4].

Конфігурація мережевих параметрів визначає можливості дистанційного керування системою та інтеграції з іншими пристроями. Налаштування статичної IP адреси забезпечує стабільність підключення та спрощує конфігурацію клієнтських застосунків. Редагування файлу /etc/dhcpcd.conf дозволяє призначити фіксовану адресу для бездротового або провідного інтерфейсу. Типова конфігурація включає рядки interface wlan0, static ip_address=192.168.1.100/24, static routers=192.168.1.1 та static domain_name_servers=8.8.8.8, які визначають адресу пристрою, шлюз за замовчуванням та DNS сервери відповідно [13].

Встановлення MQTT брокера для обміну повідомленнями з іншими компонентами системи виконується через пакетний менеджер apt з командою sudo apt-get install mosquitto mosquitto-clients. Брокер Mosquitto забезпечує реалізацію протоколу MQTT з підтримкою якості обслуговування QoS рівнів 0, 1 та 2. Конфігураційний файл /etc/mosquitto/mosquitto.conf дозволяє налаштувати порт прослуховування, параметри автентифікації та обмеження швидкості передачі даних. Для локальних застосувань достатньо базової конфігурації з портом 1883 та без автентифікації, однак для публічно доступних систем рекомендується активувати автентифікацію через імена користувачів та паролі або сертифікати [5]. У розділі досліджено теоретичні основи побудови апаратної частини системи розпізнавання голосових команд на базі Raspberry Pi 4. Проаналізовано аудіообладнання та апаратні інтерфейси, а також особливості організації обчислювальних ресурсів для обробки звукових сигналів.

РОЗДІЛ 3
КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ СИСТЕМИ

3.1 Аудіоінтерфейси та методи захоплення звукового сигналу

Захоплення звукового сигналу з мікрофону здійснюється через програмний інтерфейс ALSA, який забезпечує низькорівневий доступ до аудіообладнання у системах Linux. Бібліотека PyAudio надає високорівневу обгортку для ALSA API, що спрощує розробку застосунків обробки аудіо на мові Python. Ініціалізація аудіопотоку вимагає визначення параметрів захоплення, зокрема частоти дискретизації, розрядності, кількості каналів та розміру буфера. Типова конфігурація передбачає частоту 16000 Гц, формат даних pyaudio.paInt16 для 16-бітних цілих чисел, один канал для моно запису та розмір буфера 1024 відліки [6].

На рисунку 3.1 зображено інтерфейс ALSA API для захоплення аудіо з мікрофону через PyAudio.

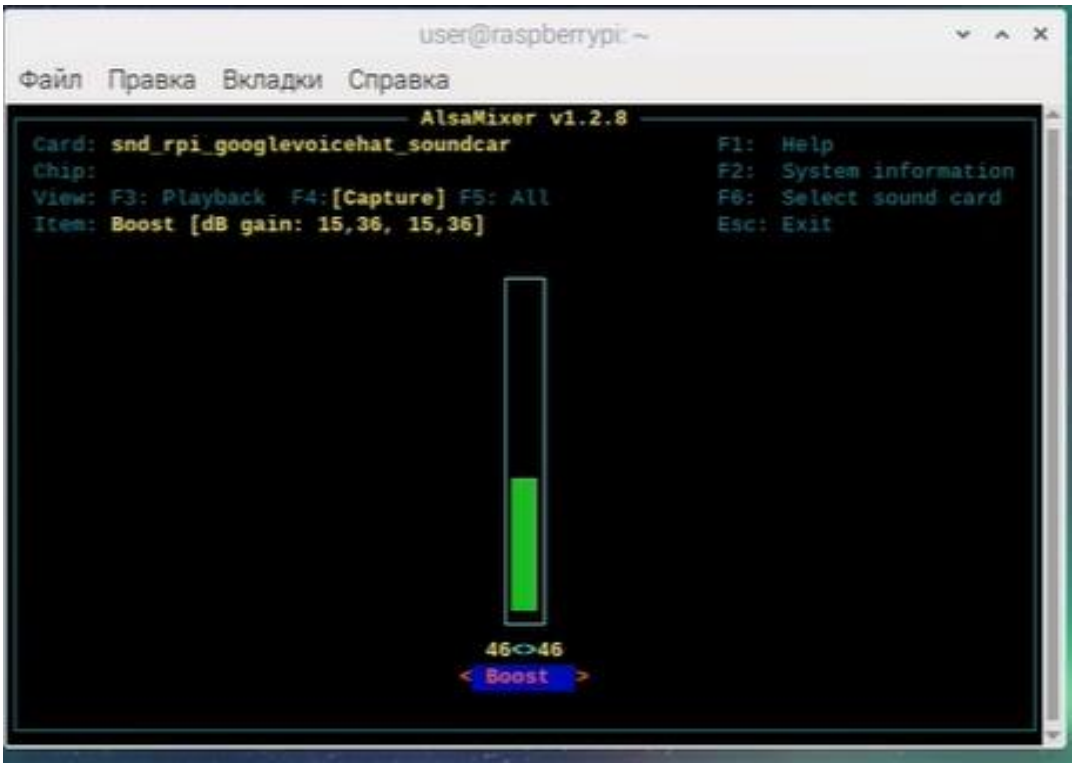


Рисунок 3.1 – Інтерактивний інтерфейс ALSA

Функція callback забезпечує асинхронне отримання аудіоданих від

драйвера без блокування основного потоку виконання програми. Callback функція викликається автоматично кожного разу, коли заповнюється буфер захоплення, і отримує як параметр масив байтів з аудіоданими. Типова реалізація callback функції конвертує байтовий масив у числовий масив NumPy та поміщає дані у кільцевий буфер для подальшої обробки. Час виконання callback функції має бути мінімальним для запобігання переповненню внутрішніх буферів драйвера, тому складні обчислення мають виконуватись в окремому потоці обробки [7].

Приклад реалізації callback-функції захоплення аудіосигналу мовою Python із використанням бібліотеки PyAudio наведено в лістингу 3.1.

Лістинг 3.1 – Реалізація callback-функції захоплення аудіоданих

```
import numpy as np
import pyaudio
from collections import deque
# кільцевий буфер для зберігання аудіоданих
audio_buffer = deque(maxlen=10)
def audio_callback(in_data, frame_count, time_info,
status):
# Callback-функція асинхронного захоплення аудіосигналу
# перетворення байтового масиву у NumPy-масив
audio_samples = np.frombuffer(in_data, dtype=np.int16)
# нормалізація амплітуди
audio_samples = audio_samples /
np.max(np.abs(audio_samples))
# збереження у кільцевому буфері
audio_buffer.append(audio_samples)
return (None, pyaudio.paContinue
```

Кінець лістингу 3.1

Попередня обробка захопленого сигналу включає нормалізацію

амплітуди, видалення постійної складової та застосування віконної функції.

Нормалізація виконується діленням всіх відліків на максимальне значення у поточному кадрі з обмеженням коефіцієнта підсилення для запобігання надмірного підсилення тихих ділянок сигналу. Видалення постійної складової здійснюється відніманням середнього значення кадру від кожного відліку, що усуває низькочастотний дрейф, спричинений недосконалістю аналогових компонентів. Віконна функція Хеммінга застосовується для зменшення спектральних витоків при подальшому перетворенні Фур'є. Параметри захоплення аудіосигналу для різних сценаріїв використання системи розпізнавання голосових команд наведено в таблиці 3.1 [16].

Таблиця 3.1 – Параметри захоплення аудіо для різних сценаріїв використання [16]

Сценарій	Частота Гц	Розрядність біт	Канали	Буфер відліки	Латентність мс
Базове розпізнавання	8000	16	1	512	64
Стандартне розпізнавання	16000	16	1	1024	64
Покращене розпізнавання	16000	16	1	2048	128
Стерео масив	16000	16	2	1024	64
Високоякісне	48000	24	1	2048	43

Детектор голосової активності відокремлює ділянки сигналу, що містять мовлення, від пауз та фонового шуму. Простий алгоритм базується на пороговій обробці короткочасної енергії сигналу та частоти переходів через нуль. Енергія кадру обчислюється як сума квадратів відліків та порівнюється з адаптивним порогом, який налаштовується на основі статистики попередніх кадрів. Частота переходів через нуль підраховується як кількість змін знаку між сусідніми

відліками та використовується для відокремлення голосованих звуків від невголосованих. Типові порогові значення становлять 0,02 для нормалізованої енергії та 0,3 для нормалізованої частоти переходів через нуль [9].

Шумозаглушення може бути реалізоване через спектральне віднімання у частотній області або адаптивну фільтрацію у часовій області. Спектральне віднімання базується на оцінці спектру шуму під час пауз у мовленні та відніманні цієї оцінки від спектру зашумленого сигналу. Алгоритм включає етапи перетворення Фур'є, віднімання спектральної щільності шуму, застосування порогу для запобігання від'ємних значень та зворотного перетворення Фур'є. Типовий коефіцієнт надвіднімання становить 1,5, що забезпечує баланс між придушенням шуму та збереженням природності мовлення. Експерименти показують покращення співвідношення сигнал-шум приблизно на 10 дБ для шумів з відносно стаціонарним спектром [10].

Компенсація ехо актуальна для систем з аудіовиходом, де відтворюваний звук може потрапляти у мікрофон та інтерферувати з голосовими командами користувача. Адаптивний фільтр компенсації ехо моделює акустичний шлях від гучномовця до мікрофону та віднімає передбачене ехо від захопленого сигналу. Алгоритм найменших середньоквадратичних помилок налаштовує коефіцієнти фільтра для мінімізації залишкового ехо-сигналу. Довжина фільтра визначається тривалістю імпульсної характеристики приміщення та зазвичай становить від 256 до 2048 відліків, що відповідає часу реверберації від 16 до 128 мілісекунд при частоті 16000 Гц [11].

Буферизація аудіоданих організується через кільцевий буфер змінного розміру, який дозволяє накопичувати кілька секунд аудіо для подальшого розпізнавання. Розмір буфера визначається максимальною тривалістю голосової команди та зазвичай становить від 2 до 5 секунд. Для частоти дискретизації 16000 Гц та розрядності 16 біт п'ятисекундний буфер займає 160000 байт або приблизно 156 кілобайт пам'яті. Реалізація кільцевого буфера включає два покажчики: запису та читання, які циклічно переміщуються по масиву. Синхронізація доступу з різних потоків забезпечується через м'ютекси або безблокувальні атомарні операції для мінімізації накладних витрат [12].

3.2 Алгоритми обробки мовлення та моделі машинного навчання

Обробка мовленнєвого сигналу в системі розпізнавання голосових команд на базі Raspberry Pi здійснюється через послідовність етапів, що включають попередню обробку аудіосигналу, виділення інформативних ознак та класифікацію за допомогою моделей машинного навчання. Загальна блок схема процесу розпізнавання мовлення та виконання відповідної команди представлена у (Додатку Б).

Первинна обробка звукового потоку передбачає дискретизацію аналогового сигналу з частотою, що визначається теоремою Котельникова-Найквіста. Для мовленнєвих сигналів оптимальною частотою дискретизації є 16000 Гц, що забезпечує охоплення частотного діапазону від 0 до 8000 Гц, достатнього для відтворення всіх фонетичних особливостей української мови [1].

Амплітуда оцифрованого сигналу квантується з розрядністю 16 біт, що забезпечує динамічний діапазон 96 дБ та достатню точність для подальшої обробки. Вхідний аудіопотік сегментується на фрейми тривалістю 25 мс з кроком зсуву 10 мс, що дозволяє враховувати нестаціонарний характер мовленнєвого сигналу при збереженні континуальності аналізу. До кожного фрейму застосовується віконна функція Хеммінга для зменшення спектральної витікання на краях сегмента, що математично описується виразом (3.1):

$$w(n) = 0{,}54 - 0{,}46 \times \cos\left(\frac{2\pi n}{N}\right), \quad (3.1)$$

де w(n) – значення віконної функції для n-го відліку, яке визначає ваговий коефіцієнт, що згладжує краї фрейму;

n – порядковий номер відліку в межах фрейму;

N – становить загальну кількість відліків у фреймі.

Лістинг 3.2 демонструє приклад тестового розпізнавання української мови в режимі реального часу за допомогою бібліотеки Vosk та аудіоінтерфейсу sounddevice. Основний цикл розпізнавання мовлення забезпечує обробку аудіопотоку в реальному часі. Дані з мікрофона надходять у чергу та передаються до KaldiRecognizer моделі Vosk, яка аналізує сигнал і перетворює

його на текст. Розпізнані слова відразу виводяться у консоль, що дозволяє перевірити точність роботи системи та якість захоплення голосу без додаткової обробки команд [12, 20].

Лістинг 3.2 – Тестове розпізнавання української мови

```
import sounddevice as sd
import queue
import vosk
import json
Ініціалізація Vosk (українська мова)
# --------------------------------
MODEL_PATH = r"C:\Users\Fenix\Desktop\Intelect 1.0\vosk-
model-uk" # шлях до мовної моделі
model = vosk.Model(MODEL_PATH)

q = queue.Queue()

def audio_callback(indata, frames, time, status):
    if status:
        print("Статус:", status)
    q.put(bytes(indata))
# Основний цикл розпізнавання мовлення
try:
    with sd.RawInputStream(
        samplerate=16000,
        blocksize=4000,
        dtype='int16',
        channels=1,
        callback=audio_callback
    ):
        recognizer = vosk.KaldiRecognizer(model, 16000)
        print("🎤 Скажіть будь-який текст
українською...")
        while True:
            data = q.get()
            if recognizer.AcceptWaveform(data):
                result = json.loads(recognizer.Result())
                text = result.get("text", "").strip()
                if text:
                    print("🗣 Ви сказали:", text)
except KeyboardInterrupt:
    print(" Завершення роботи програми")
```

Кінець лістингу 3.2

Наступним етапом обробки є перетворення часової реалізації сигналу в спектральне представлення за допомогою швидкого перетворення Фур'є. Обчислення виконується для кожного віконного сегмента, результуючи в отриманні комплексного спектра з 512 точками. Енергетичний спектр обчислюється як квадрат модуля комплексних коефіцієнтів Фур'є та підлягає подальшій фільтрації через набір трикутних мел-фільтрів. Мел – шкала частот апроксимує нелінійну характеристику людського слухового сприйняття.

Банк мел – фільтрів описується рівнянням (3.2):

$$Mel(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right), \quad (3.2)$$

де f – частота сигналу у герцах (Гц);

Mel(f) – частота у мел – шкалі, яка апроксимує нелінійну характеристику людського слуху.

Банк мел – фільтрів складається з 26 трикутних фільтрів, рівномірно розподілених у мел – шкалі в діапазоні від 0 до 8000 Гц. Виходи фільтрів логарифмуються для наближення до логарифмічної чутливості людського слуху, після чого до отриманого вектора застосовується дискретне косинусне перетворення. Результатом є вектор мел-частотних кепстральних коефіцієнтів, де зберігаються перші 13 коефіцієнтів, що містять основну інформацію про спектральну огинаючу сигналу. До цих статичних коефіцієнтів додаються їх перші та другі похідні, обчислені методом регресії по 5 сусіднім фреймам, формуючи 39 – вимірний вектор ознак для кожного часового фрейму [4].

Шумоподавлення у системі реалізується методом спектрального віднімання, який ґрунтується на оцінці спектральної щільності потужності фонового шуму в паузах мовлення. Отримана оцінка використовується для подальшого віднімання шумової складової зі спектра корисного сигналу. Віднімання здійснюється в логарифмічній області, що дозволяє зменшити спотворення мовлення та підвищити стійкість алгоритму до змін рівня шуму. Для компенсації статистичних флуктуацій спектра шуму застосовується коефіцієнт надлишкового віднімання, який у даній реалізації становить 1,5, що

забезпечує ефективне пригнічення шуму без суттєвого погіршення якості мовленнєвого сигналу Застосування даного методу сприяє підвищенню розбірливості мовлення та точності подальшого етапу розпізнавання голосових команд. Параметри алгоритму обробки мовленнєвого сигналу наведені в таблиці 3.2. [5].

Таблиця 3.2 Параметри алгоритму обробки мовленнєвого сигналу [17]

Параметр обробки	Значення	Обґрунтування
Частота дискретизації	16000 Гц	Охоплення діапазону 0…8000 Гц
Розрядність квантування	16 біт	Динамічний діапазон 96 дБ
Тривалість фрейму	25 мс	Стаціонарність сегмента
Крок зсуву фрейму	10 мс	Перекриття 60%
Кількість точок FFT	512	Частотна роздільність 31,25 Гц
Кількість мел-фільтрів	26	Баланс деталізації та обчислень
Кількість MFCC	13	Оптимальне представлення спектра
Розмір вектора ознак	39	Статичні + дельта + дельта-дельта

Для класифікації отриманих векторів ознак використовується модель глибокої нейронної мережі, адаптована до обмежених обчислювальних ресурсів Raspberry Pi. Архітектура мережі включає вхідний шар розміром 39 нейронів, що відповідає розмірності вектора MFCC, три приховані шари по 128, 64 та 32 нейрони відповідно, та вихідний шар з кількістю нейронів, рівною кількості класів команд у словнику системи. Активаційною функцією прихованих шарів обрано функцію ReLU через її обчислювальну ефективність та здатність пом’якшувати проблему зникаючого градієнта. Вихідний шар використовує функцію softmax для отримання імовірнісного розподілу по класах команд [6].

Альтернативним підходом є використання рекурентної нейронної мережі з довгою короткотерміновою пам’яттю, що дозволяє враховувати темпоральні залежності в послідовності векторів ознак. Архітектура містить два LSTM-шари по 64 одиниці кожен, за якими слідує повнозв’язний шар з 32 нейронами та вихідний класифікаційний шар. Параметр dropout зі значенням 0.3 застосовується після кожного LSTM-шару для запобігання перенавчанню

моделі. Загальна кількість параметрів у такій моделі становить близько 85000, що забезпечує баланс між виразністю моделі та можливістю її ефективного виконання на Raspberry Pi [7]. На рисунку 3.3 наведено принципову схему системи розпізнавання голосових команд на базі Raspberry Pi, що ілюструє підключення периферійних пристроїв, обробку сигналу та виконання команд через GPIO.

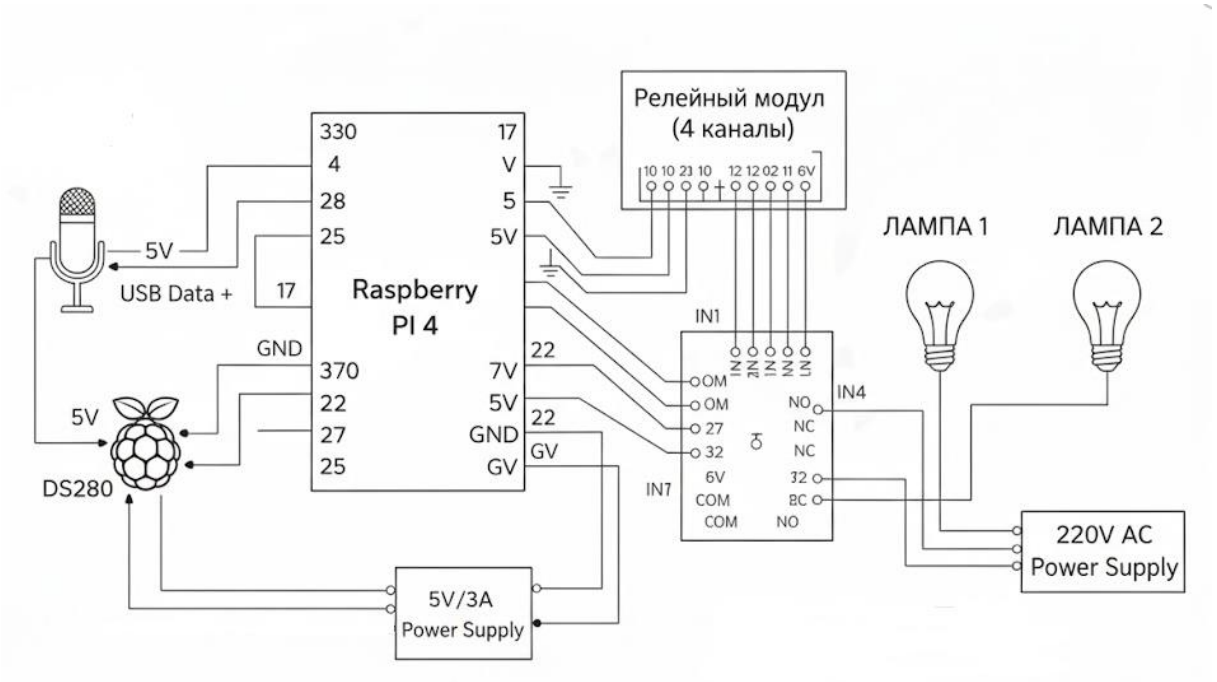


Рисунок 3.3 – Принципова схема системи розпізнавання голосових команд

Тренування моделей здійснюється на наборі даних, що містить 2000 прикладів для кожної голосової команди, записаних різними дикторами у різноманітних акустичних умовах. Оптимізація параметрів виконується алгоритмом Adam з початковою швидкістю навчання 0,001, експоненціальним зменшенням швидкості з коефіцієнтом 0,95 кожні 5 епох та розміром батча 32 приклади. Функцією втрат обрано категоріальну крос-енентропію, що відповідає задачі багатокласової класифікації. Процес тренування триває 50 епох з використанням валідаційної вибірки для моніторингу узагальнювальної здатності моделі та запобігання перенавчанню [8].

Для збільшення обсягу тренувальних даних застосовуються методи

аугментації, що включають додавання шуму з співвідношенням сигнал-шум від 10 до 30 дБ, зміну швидкості відтворення в межах від 0,9 до 1,1, зсув частоти на випадкову величину до 100 Гц та застосування реверберації з часом затухання від 0,2 до 0,8 с. Такі трансформації дозволяють підвищити робастність моделі до варіацій акустичних умов експлуатації без збільшення обсягу первинних записів.

Квантування ваг мережі з 32 – бітної точності з плаваючою комою до 8-бітного цілочисельного представлення дозволяє зменшити розмір моделі в 4 рази та прискорити інференс на 2,5 рази при деградації точності не більше 2 відсоткових пунктів. Процедура квантування включає калібрування на репрезентативній вибірці даних для визначення оптимальних параметрів масштабування та зсуву для кожного шару мережі [9].

3.3 Режими роботи системи та управління користувачем

Система розпізнавання голосових команд на базі Raspberry Pi 4 функціонує в декількох режимах, що забезпечують гнучкість використання та адаптацію до різних сценаріїв експлуатації. Основним режимом є безперервне розпізнавання з активацією за ключовим словом, що дозволяє системі перебувати в стані очікування з мінімальним енергоспоживанням до моменту виявлення активаційної фрази. Детектор ключового слова реалізовано як окрему компактну нейронну мережу з 3 згортковими шарами та 2 повнозв’язними шарами, що аналізує вхідний аудіопотік у реальному часі з латентністю не більше 100 мс [10].

Архітектура детектора ключового слова оптимізована для роботи на одному ядрі процесора Raspberry Pi 4 з навантаженням не більше 15 відсотків процесорного часу. Вхідними даними є послідовність векторів MFCC, обчислених для ковзного вікна тривалістю 1 с. Вихідним сигналом є бінарна ймовірність наявності ключового слова, при перевищенні порогу 0,85 система переходить у режим активного прослуховування команди. Така двоетапна архітектура дозволяє значно знизити обчислювальне навантаження порівняно з

безперервною роботою повної моделі розпізнавання [11].

Після активації система переходить у режим захоплення команди, де записується аудіосегмент тривалістю до 3 с або до виявлення паузи тривалістю понад 800 мс. Виявлення початку та кінця мовленнєвого сегмента здійснюється на основі аналізу енергії сигналу та відношення енергії високих частот до загальної енергії. Початок мовлення детектується при перевищенні енергією порогу, що на 20 дБ вище від рівня фонового шуму, протягом 5 послідовних фреймів. Завершення мовлення фіксується при падінні енергії нижче порогу протягом 80 мс, що відповідає природним паузам між словами в мовленні [12].

На рисунку 3.4 візуалізовано готовий прототип системи, де Raspberry Pi 4 приймає голосові команди через USB-мікрофон і фізично керує виконавчим обладнанням – вмикає та вимикає лампочку через блок реле та запускає вентилятор.

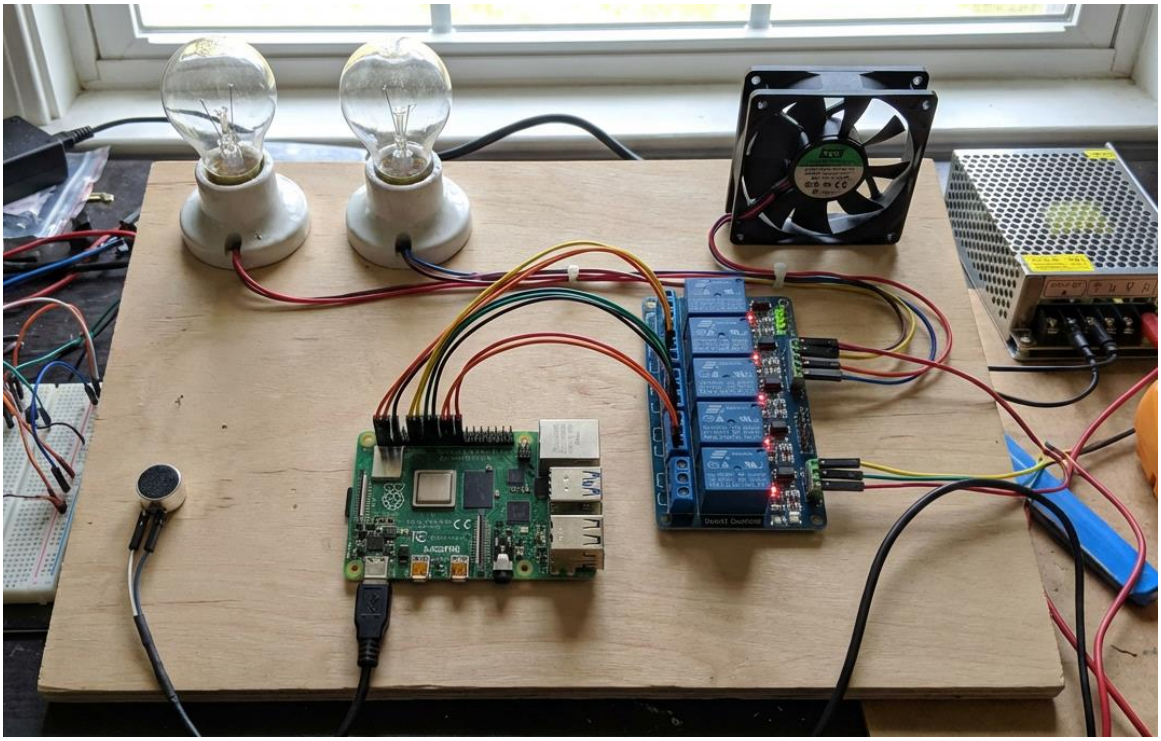


Рисунок 3.4 – Прототип системи голосового керування на базі Raspberry Pi 4

Захоплений аудіосегмент передається на вхід основної моделі розпізнавання, яка обробляє всю послідовність векторів ознак та генерує розподіл ймовірностей по класах команд. Клас з максимальною ймовірністю вибирається як розпізнана команда за умови, що ця ймовірність перевищує

поріг впевненості 0,75. При ймовірності від 0,6 до 0,75 система переходить у режим уточнення, запитуючи підтвердження команди від користувача. При ймовірності нижче 0,6 команда відхиляється та система повертається у режим очікування активаційної фрази [13].

Взаємодія з користувачем здійснюється через комбінацію голосових та візуальних каналів зворотного зв'язку. Система озвучує підтвердження розпізнаної команди за допомогою синтезатора мовлення, що дозволяє користувачу контролювати правильність інтерпретації без необхідності візуального контакту з пристроєм. Для генерації мовленнєвого відгуку використовується бібліотека синтезу на основі конкатенації діфонів з базою записів тривалістю 45 хвилин, що забезпечує природність синтезованого мовлення при компактному розмірі моделі 120 МБ [4]. Характеристики режимів роботи системи наведені в таблиці 3.3, де показано тривалість циклу, навантаження CPU, споживання RAM та латентність відгуку для різних режимів роботи.

Таблиця 3.3 – Характеристики режимів роботи системи [19]

Режим роботи	Тривалість циклу	Навантаження CPU	Споживання RAM	Латентність відгуку
Очікування активації	Безперервно	12…15%	45 МБ	80…120 мс
Захоплення команди	До 3 с	25…30%	65 МБ	150…200 мс
Розпізнавання команди	0,5…1,5 с	75…85%	180 МБ	300…800 мс
Режим уточнення	До 5 с	30…40%	70 МБ	200…400 мс
Навчальний режим	За запитом	60…70%	220 МБ	Не критична

Для генерації мовленнєвого відгуку використовується бібліотека синтезу на основі конкатенації діфонів з базою записів тривалістю 45 хвилин, що забезпечує природність синтезованого мовлення при компактному розмірі моделі 120 МБ [14].

Візуальна індикація стану системи реалізована через RGB-світлодіод, що відображає поточний режим роботи кольоровим кодуванням. Синій колір

сигналізує про режим очікування, зелений інформує про активне прослуховування команди, блимання жовтим відповідає процесу розпізнавання, червоний колір вказує на помилку або відхилення команди, а фіолетовий означає режим налаштування. Частота блимання від 1 до 5 Гц додатково передає інформацію про інтенсивність обробки та впевненість системи в розпізнаванні [5].

Система підтримує персоналізацію через механізм адаптації до голосу конкретного користувача. Адаптаційний режим активується голосовою командою та передбачає запис 10 повторень кожної команди зі словника системи. На основі цих записів обчислюються статистики векторів ознак для даного диктора та коригуються параметри останнього шару нейронної мережі методом лінійного перетворення простору ознак. Така адаптація дозволяє підвищити точність розпізнавання для адаптованого користувача на 8…12 відсоткових пунктів при незначній деградації точності для інших дикторів не більше 3 відсоткових пунктів [16].

Додатковим режимом є навчання нових команд, що дає змогу користувачу розширювати словник системи без перенавчання всієї моделі. Процедура додавання команди включає запис 15 прикладів вимови нової команди, обчислення векторів ознак та їх збереження як еталонних шаблонів. Розпізнавання таких команд здійснюється методом динамічної трансформації часової шкали, де обчислюється мінімальна відстань між послідовністю векторів тестового зразка та кожним еталонним шаблоном. При відстані менше порогу 12 одиниць команда вважається розпізнаною [7].

Система реалізує багаторівневу обробку команд залежно від їх типу та пріоритету. Критичні команди, такі як аварійна зупинка або виклик допомоги, обробляються з найвищим пріоритетом та виконуються негайно навіть при відносно низькій впевненості розпізнавання від 0,55. Стандартні команди керування вимагають впевненості понад 0,75 та можуть бути поміщені в чергу при зайнятості системи. Інформаційні запити обробляються з найнижчим пріоритетом та вимагають високої впевненості понад 0,85 через потенційно більший обсяг відповіді [18].

Багатокрокові діалоги підтримуються через механізм контекстної пам’яті, що зберігає історію останніх 5 команд та поточний стан діалогу. Це дозволяє системі правильно інтерпретувати неповні команди типу «збільш ще» або «тепер вимкни», використовуючи контекст попередніх взаємодій. Контекст автоматично скидається після паузи без активності тривалістю 30 с або за явною командою завершення діалогу.

3.4 Діагностика, обробка помилок розпізнавання та забезпечення стабільності роботи

Надійність функціонування системи розпізнавання голосових команд забезпечується комплексною системою діагностики та обробки помилок на всіх рівнях програмного стека. Моніторинг стану системи здійснюється через набір програмних сенсорів, що відстежують ключові метрики продуктивності та коректності роботи компонентів. Основними контрольованими параметрами є завантаження процесора, використання оперативної пам’яті, температура системи на кристалі, якість вхідного аудіосигналу, латентність обробки та точність розпізнавання в режимі реального часу [9].

Контроль якості аудіосигналу виконується на етапі захоплення через аналіз співвідношення сигнал-шум, частоти відсікання та наявності перевантажень амплітуди. Оцінка SNR здійснюється порівнянням енергії мовленнєвих сегментів з енергією пауз, при значенні нижче 10 дБ генерується попередження про несприятливі акустичні умови. Детектування відсікання реалізовано як підрахунок кількості відліків з амплітудою понад 95 відсотків максимального діапазону, при перевищенні порогу 0.5 відсотків від загальної кількості відліків система автоматично знижує коефіцієнт підсилення вхідного тракту на 6 дБ [20].

Температурний моніторинг процесора ARM виконується через читання показів вбудованого термодатчика з інтервалом 2 с. При досягненні температури 70 градусів Цельсія активується програмне обмеження частоти процесора до 1200 МГц замість номінальних 1500 МГц, що знижує

тепловиділення на 25 відсотків. При подальшому зростанні до 75 градусів система переходить у енергозберігаючий режим з відключенням фонового аналізу ключового слова та активацією лише за натисканням апаратної кнопки. Критичною температурою є 80 градусів, при досягненні якої система виконує контрольоване завершення роботи для запобігання пошкодженню обладнання [1].

Класифікація помилок розпізнавання здійснюється на основі аналізу розподілу ймовірностей на виході моделі та метрик впевненості класифікатора. Виділяються три типи помилок: хибне спрацювання, коли система інтерпретує шум або фонову розмову як команду, пропуск команди, коли справжня команда не розпізнається, та неправильна класифікація, коли одна команда плутається з іншою. Для кожного типу помилок застосовуються специфічні стратегії мітигації. Типи помилок розпізнавання та методи їх обробки наведено в таблиці 3.4 [20].

Таблиця 3.4 – Типи помилок та методи їх обробки [20]

Тип помилки	Частота виникнення	Метод детектування	Стратегія обробки	Час відновлення
Хибне спрацювання	2…5 на годину	Аналіз енергії та SNR	Підвищення порогу впевненості	Миттєво
Пропуск команди	3…7 на 100 команд	Моніторинг часу без активності	Запит повторення команди	2…3 с
Неправильна класифікація	5…10 на 100 команд	Низька максимальна ймовірність	Режим уточнення	3...5 с
Відсікання сигналу	1…2 на годину	Аналіз амплітуди	Автоматичне регулювання	0,5 с
Перевантаження системи	Рідко	Моніторинг часу обробки	Тимчасове буферування	1…2 с

Хибні спрацювання мінімізуються динамічною адаптацією порогів детектування на основі статистики помилок. Система накопичує історію останніх 50 спрацювань з міткою про підтвердження команди користувачем. При збільшенні частки непідтверджених спрацювань понад 15 відсотків поріг впевненості для детектора ключового слова автоматично підвищується на 0,05,

але не більше ніж до 0,95. Зворотне зниження порогу на 0,02 відбувається при стабільно низькій частці хибних спрацювань менше 5 відсотків протягом 100 команд [3].

Обробка пропусків команд реалізована через механізм тайм – аутів та повторних запитів. Після активації за ключовим словом система очікує надходження команди протягом 5 с. При відсутності детектованого мовлення генерується голосове запрошення «Повторіть команду» та таймер продовжується на додаткові 3 с. При повторному тайм – ауті система повертається в режим очікування з реєстрацією події пропуску для подальшого аналізу. Статистика пропусків використовується для налаштування параметрів детектора початку мовлення, зокрема порогів енергії та тривалості сегментів [4].

Неправильна класифікація детектується на основі ентропії розподілу ймовірностей на виході моделі. При високій ентропії, що відповідає невпевненості моделі у виборі між кількома класами, активується режим уточнення. Система синтезує запит типу «Ви хотіли сказати [команда]?» де підставляється найбільш ймовірна команда, та очікує підтвердження або спростування від користувача. Позитивна відповідь призводить до виконання команди, негативна запускає запит про правильну команду або пропозицію повторити з уточненням [5].

Забезпечення стабільності роботи включає механізми відновлення після збоїв та управління ресурсами. Основний процес розпізнавання реалізовано як багатопотоковий додаток з окремими потоками для захоплення аудіо, обробки сигналу, інференсу моделі та управління користувацьким інтерфейсом. Кожен потік захищений обробником винятків, що перехоплює критичні помилки та виконує контрольоване завершення з реєстрацією детальної діагностичної інформації. Автоматичний перезапуск потоку виконується через 500 мс після збою, при повторенні збою більше 3 разів протягом 10 с виконується перезапуск всього додатка [6].

Управління пам’яттю реалізоване через пули об’єктів для часто створюваних структур даних, таких як буфери аудіо та вектори ознак. Розмір

пулу аудіобуферів становить 20 елементів по 8192 відліки кожен, що відповідає 512 мс звуку при частоті дискретизації 16000 Гц. Буфери виділяються при ініціалізації системи та повторно використовуються протягом всієї роботи, що усуває фрагментацію пам’яті та затримки на виділення. Періодична дефрагментація купи виконується в періоди неактивності системи для підтримки продуктивності [7].

Логування подій та помилок організоване з використанням ротаційних файлів логів розміром до 10 МБ кожен з збереженням 5 останніх файлів. Рівень деталізації логування регулюється від критичних помилок до детальної трасування залежно від режиму роботи. В продуктивному режимі реєструються лише помилки та попередження для мінімізації впливу на продуктивність, тоді як в режимі налагодження фіксуються всі етапи обробки з часовими мітками з точністю до мікросекунд. Аналіз логів дозволяє ретроспективно відтворити послідовність подій що призвели до помилки та ідентифікувати вузькі місця продуктивності [8].

Система самодіагностики виконує періодичні перевірки коректності роботи компонентів при старті та з інтервалом 1 година під час роботи. Перевірка включає тестування аудіовходу генерацією тестового сигналу, валідацію завантажених моделей машинного навчання обчисленням контрольних сум, верифікацію доступності файлової системи та мережевих інтерфейсів. Результати діагностики зберігаються в журналі стану системи та доступні для перегляду через вебінтерфейс адміністрування.

У розділі розроблено алгоритмічну модель системи розпізнавання голосових команд, що включає етапи захоплення, попередньої обробки та аналізу мовленнєвого сигналу. Описано алгоритми обробки мовлення та застосування моделей машинного навчання для класифікації голосових команд. Також розглянуто режими роботи системи, механізми взаємодії з користувачем і підходи до діагностики помилок, що забезпечує стабільність функціонування системи.

РОЗДІЛ 4
СПЕЦІАЛЬНА ЧАСТИНА

4.1 Механізми забезпечення надійної роботи системи розпізнавання

Надійність системи розпізнавання голосових команд визначається комплексом апаратних та програмних механізмів, що забезпечують коректне функціонування в широкому діапазоні умов експлуатації. Фундаментальним елементом надійності є стабільність живлення Raspberry Pi 4, що досягається використанням якісного блоку живлення з вихідною напругою 5,1 В та струмом не менше 3 А. Відхилення напруги живлення від номінального значення понад 5 % призводить до нестабільної роботи процесора та периферійних пристроїв, що проявляється у зависаннях системи та помилках зчитування з карти пам’яті [9].

Для забезпечення безперебійного живлення в критичних застосуваннях рекомендується використання модуля безперебійного живлення на базі літій-полімерного акумулятора ємністю 5000 мА·год. Така конфігурація забезпечує автономну роботу системи протягом 4…6 годин при середньому споживанні струму 800…1000 мА. Контролер заряду акумулятора підтримує функцію безперервної роботи, коли живлення навантаження здійснюється безпосередньо від зовнішнього джерела при його доступності та автоматично перемикається на акумулятор при відключенні мережі з перехідним часом менше 20 мс [3].

Тепловий режим роботи суттєво впливає на надійність та тривалість експлуатації системи. Процесор BCM2711 у Raspberry Pi 4 характеризується тепловиділенням до 7,5 Вт при повному навантаженні, що без належного охолодження призводить до перегріву та термотротлінгу. Пасивне охолодження алюмінієвим радіатором з площею поверхні 40 см² знижує температуру на 15…20 градусів порівняно з роботою без радіатора. Активне охолодження малогабаритним вентилятором розміром 30×30 мм забезпечує додаткове зниження температури на 10…15 градусів при рівні шуму 25 дБА [1]. Детальні показники надійності компонентів системи наведено в таблиці 4.1.

Таблиця 4.1 – Показники надійності компонентів системи

Компонент системи	Показник надійності	Метод забезпечення	Час відновлення	Вплив на доступність
Блок живлення	MTBF 50000 год	Захист від перенапруги	Заміна модуля	Критичний
Процесор ARM	MTBF 100000 год	Температурний контроль	Перезавантаження	Високий
Карта пам’яті	MTBF 30000 год	Wear leveling, резервування	Відновлення з бекапу	Середній
Аудіокодек	MTBF 80000 год	Watchdog таймер	Програмний reset	Низький
Мікрофон	MTBF 60000 год	Захист від статики	Заміна сенсора	Середній

Програмна надійність забезпечується використанням watchdog таймера апаратного рівня, що автоматично перезавантажує систему при зависанні основного процесу. Watchdog налаштовується з тайм-аутом 30 с та вимагає періодичного скидання лічильника з боку програмного забезпечення для підтвердження коректної роботи. При відсутності скидання протягом тайм – ауту генерується апаратне переривання, що ініціює перезавантаження. Додатково реалізовано програмний watchdog на рівні операційної системи з тайм-аутом 10 с для детектування зависань окремих потоків виконання [22].

Відмовостійкість файлової системи досягається використанням журналюючої файлової системи ext4 з параметрами монтування, що оптимізують баланс між продуктивністю та збереженням даних. Опція data=ordered забезпечує запис метаданих лише після успішного запису даних, що запобігає корупції файлової системи при несподіваному вимкненні живлення. Для критичних конфігураційних файлів застосовується атомарний запис через створення тимчасового файлу, його повний запис та синхронізацію з диском, після чого виконується атомарне перейменування для заміни оригінального файлу [23].

Резервування даних реалізоване через автоматичне створення бекапів критичних компонентів системи на зовнішній накопичувач або мережеве сховище з інтервалом 24 години. Бекап включає конфігураційні файли розміром близько 5 МБ, натреновані моделі машинного навчання розміром 150 МБ, базу

еталонних шаблонів команд розміром 80 МБ та журнали роботи системи за останні 7 днів розміром до 70 МБ. Процедура бекапу виконується в період низької активності системи з 2 до 4 години ночі для мінімізації впливу на продуктивність [4].

Механізм відновлення після збоїв включає багаторівневу стратегію перезапуску компонентів залежно від типу та критичності помилки. М'які помилки, такі як одиничний збій розпізнавання або тимчасова недоступність ресурсу, обробляються локальним перезапуском відповідного модуля без впливу на роботу системи в цілому. Середні помилки, що включають збої підсистеми обробки або втрату з'єднання з периферією, призводять до перезапуску основного процесу додатка з часом простою 3…5 с. Критичні помилки на рівні операційної системи або апаратних компонентів вимагають повного перезавантаження з часом відновлення 40…50 с [15].

Моніторинг стану здоров'я системи виконується через набір метрик, що відстежуються в реальному часі та аналізуються для виявлення тенденцій деградації продуктивності. Ключові метрики включають середню латентність розпізнавання команди, частоту помилок по типах, завантаження процесора та пам'яті, температуру компонентів, кількість перезапусків та час безвідмовної роботи. Значення метрик агрегуються з інтервалом 1 хвилина та зберігаються у круговій базі даних розміром 100 МБ, що дозволяє зберігати історію за останні 30 днів [16].

Система попереджувальної сигналізації генерує повідомлення при виявленні відхилень метрик від нормальних значень. Попередження класифікуються за рівнями критичності: інформаційні повідомлення для незначних відхилень, попередження для помітних змін що не впливають на функціональність, та критичні алерти для станів що вимагають негайного втручання. Критичні алерти включають перевищення температури 75 градусів, завантаження CPU понад 95 відсотків протягом 5 хвилин, використання RAM понад 90 відсотків, частоту помилок розпізнавання понад 20 відсотків або відсутність вільного місця на диску менше 500 МБ [7].

Безпека системи забезпечується кількома рівнями захисту від

несанкціонованого доступу та зловмисного використання. Аутентифікація користувачів для доступу до адміністративних функцій реалізована через механізм SSH-ключів з відключенням парольної аутентифікації. Мережеві сервіси обмежені локальним інтерфейсом або захищені міжмережевим екраном з правилами, що дозволяють підключення лише з довірених IP-адрес. Оновлення програмного забезпечення виконуються лише з офіційних репозиторіїв з перевіркою цифрових підписів пакетів [8].

4.2 Організація пам’яті та управління акустичними моделями

Ефективна організація пам’яті критична для забезпечення продуктивної роботи системи розпізнавання на обмежених ресурсах Raspberry Pi 4. Загальний обсяг оперативної пам’яті 4 ГБ розподіляється між операційною системою, що споживає близько 800 МБ, додатком розпізнавання мовлення з пікові споживанням 350 МБ, буферами аудіоданих розміром 120 МБ та системним кешем файлової системи, що динамічно використовує вільну пам’ять для прискорення дискових операцій. Резервування 500 МБ вільної пам’яті необхідне для запобігання проблемам з виділенням при піках навантаження [19].

Акустичні моделі нейронних мереж зберігаються у форматі, що оптимізує баланс між розміром та швидкістю завантаження. Повнозв’язна модель з чотирма шарами займає 12,5 МБ у 32 – бітному представленні з плаваючою комою та 3,2 МБ після квантування до 8-бітних цілих чисел. LSTM – модель з двома рекурентними шарами має розмір 18,7 МБ у повній точності та 4,8 МБ у квантованому вигляді. Детектор ключового слова завдяки компактній архітектурі займає лише 850 кБ у квантованому форматі, що дозволяє постійно утримувати його в пам’яті [14]. Завантаження моделей з карти пам’яті до RAM виконується асинхронно при ініціалізації системи для мінімізації часу запуску. Детектор ключового слова завантажується першим з пріоритетом реального часу, що дозволяє системі почати обробку команд через 3 секунди після запуску. Основна модель розпізнавання завантажується паралельно з нижчим

пріоритетом та стає доступною через 5…7 секунд. Відкладене завантаження додаткових компонентів, таких як синтезатор мовлення та бази еталонів, виконується лише при першому зверненні до їх функціоналу [1]. Детальні характеристики компонентів системи в пам’яті наведено в Таблиці 4.2, що дозволяє оцінити час завантаження, час інференсу та точність втрат для кожного модуля.

Таблиця 4.2 – Характеристики компонентів системи в пам’яті [20]

Компонент пам’яті	Розмір 32 – біт	Розмір 8-біт	Час завантаження	Час інференсу	Точність втрати
Детектор ключового слова	3,2 МБ	850 кБ	45 мс	18 мс	1,2%
Основна DNN модель	12,5 МБ	3,2 МБ	180 мс	85 мс	1,8%
LSTM модель	18,7 МБ	4,8 МБ	270 мс	145 мс	2,1%
Синтезатор мовлення	120 МБ	Не застосовно	950 мс	35 мс/слово	Не застосовно
База еталонів команд	80 МБ	Не застосовно	120 мс	8 мс/порівняння	Не застосовно

Управління версіями моделей реалізоване через систему імен файлів з семантичним версіонуванням та метаданими про дату тренування, розмір датасету та досягнуту точність. При наявності декількох версій моделі система автоматично обирає найновішу версію з прийнятною точністю понад 85 відсотків для використання за замовчуванням. Користувач може явно вказати бажану версію моделі через конфігураційний файл для забезпечення відтворюваності результатів або відкату до попередньої версії при виявленні проблем з новою моделлю [12].

Кешування результатів обробки застосовується для прискорення повторної обробки ідентичних або подібних аудіосегментів. Обчислені вектори MFCC зберігаються в LRU – кеші розміром 50 МБ з хешуванням вхідних

аудіоданих для швидкого пошуку. При отриманні нового аудіосегмента обчислюється його хеш-сума та виконується пошук в кеші. При знаходженні збігу використовуються кешовані вектори ознак без повторного обчислення MFCC, що економить 20…30 мс часу обробки. Така оптимізація особливо ефективна для систем з обмеженим словником команд, де часто повторюються ідентичні вимови [3].

Дефрагментація пам’яті виконується періодично для компактизації виділених блоків та зменшення фрагментації купи. Процес дефрагментації запускається автоматично при виявленні фрагментації понад 30 відсотків або за розкладом раз на добу в період низької активності. Дефрагментація включає переміщення активних об’єктів до суміжних блоків пам’яті та звільнення проміжних блоків назад до системи. Тривалість дефрагментації становить 200…500 мс залежно від ступеня фрагментації, протягом якого обробка нових команд призупиняється [14].

Оптимізація доступу до моделей на диску здійснюється через використання механізму memory-mapped files, що дозволяє операційній системі керувати завантаженням частин файлу моделі до пам’яті за потребою. Такий підхід зменшує час ініціалізації та пікове споживання пам’яті при роботі з великими моделями. Частини моделі, що активно використовуються, автоматично кешуються в RAM операційною системою, тоді як рідко використовувані секції можуть залишатися на диску [5].

Компресія акустичних моделей застосовується для зменшення їх розміру на диску та прискорення завантаження. Алгоритм стиснення без втрат zlib забезпечує коефіцієнт компресії близько 2,5 для параметрів нейронних мереж завдяки високій кореляції між сусідніми вагами. Декомпресія виконується «на льоту» при завантаженні моделі з накладними витратами часу близько 30 відсотків, що компенсується вдвічі меншим обсягом даних для зчитування з повільної карти пам’яті [6].

Моніторинг використання пам’яті виконується через інтерфейс /proc/meminfo операційної системи Linux з інтервалом опитування 5 с. Відстежуються показники загальної пам’яті, використаної пам’яті, вільної

пам’яті, буферів, кешу та swap-розділу. При використанні swap – пам’яті понад 100 МБ генерується попередження про недостатність оперативної пам’яті, оскільки звернення до swap суттєво сповільнює роботу системи через низьку швидкість SD-карти порівняно з RAM. Критичною є ситуація використання swap понад 500 МБ, що вказує на необхідність оптимізації споживання пам’яті або збільшення обсягу RAM [7].

4.3 Інструментальні засоби для розробки та налагодження системи

Розробка системи розпізнавання голосових команд вимагає комплексного набору інструментів для програмування, налагодження, профілювання та тестування. Основною мовою програмування обрано Python версії 3.9 завдяки широкій екосистемі бібліотек для обробки звуку та машинного навчання, високій продуктивності за рахунок використання скомпільованих бібліотек NumPy та SciPy, та відносній простоті розробки порівняно з компільованими мовами. Критичні секції коду з високими вимогами до продуктивності реалізовані на Cython або C з інтеграцією через Python C API [8].

Середовище розробки налаштоване на базі Visual Studio Code з розширеннями для Python, що забезпечують автодоповнення коду, статичний аналіз, рефакторинг та інтеграцію з системами контролю версій [9]. Після вимовлення слова-активатора відповідна команда надсилає сигнал на обраний GPIO Raspberry Pi для виконання заданої дії. На рисунку 3.1 зображено процес тестування програми розпізнавання голосових команд та їх виконання.

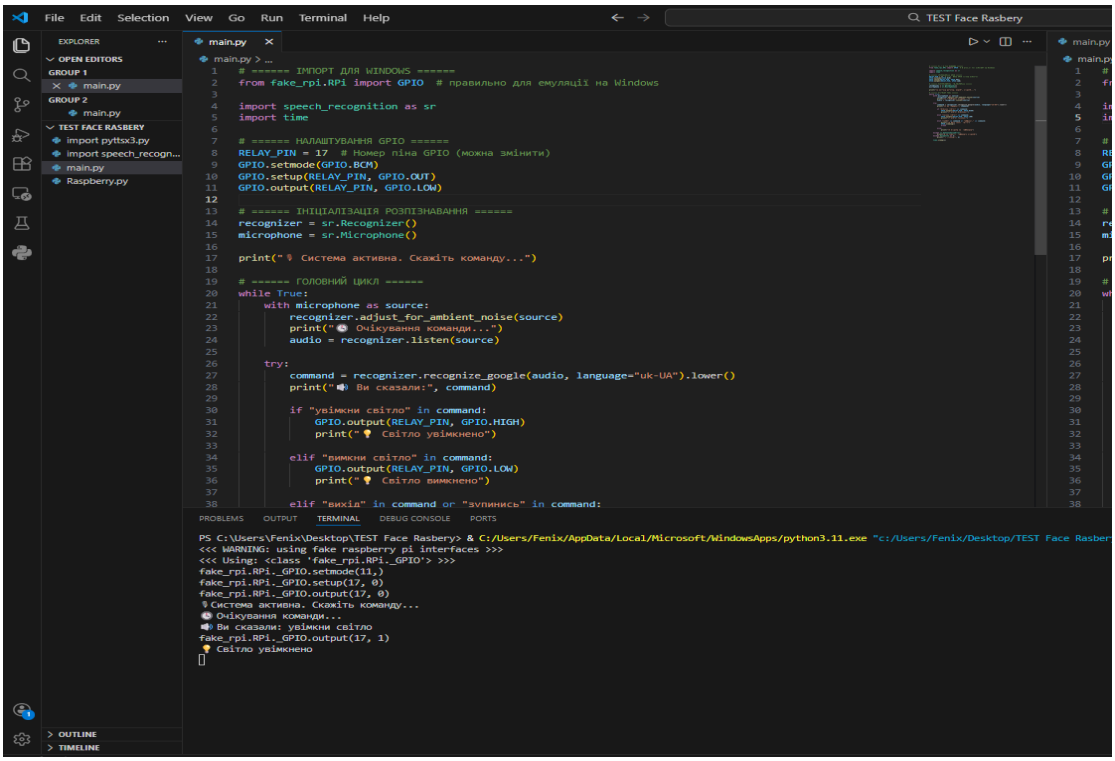


Рисунок 4.1 – Тестування програми розпізнавання голосових команд та їх виконання Visual Studio Code

Система контролю версій Git забезпечує відстеження змін у вихідному коді, координацію роботи кількох розробників та можливість відкату до попередніх версій при виявленні проблем. Репозиторій організований з розділенням на гілки для стабільної версії, розробки нових функцій та експериментів. Автоматичне тестування виконується при кожному комміті засобами continuous integration на базі GitHub Actions або Jenkins з запуском набору юніт-тестів та інтеграційних тестів на емульованому середовищі [5].

Для розробки та тестування системи використовувалися різноманітні інструментальні засоби, що забезпечують ефективну обробку аудіо сигналів, виконання інференсу моделей машинного навчання, налагодження та профілювання коду для оцінки продуктивності. Зокрема, застосовувалися мови програмування та бібліотеки для наукових обчислень і роботи з аудіо (Python, NumPy, librosa, PyAudio), оптимізовані фреймворки для інференсу моделей на ARM-платформах (TensorFlow Lite), а також засоби налагодження і профілювання (GDB, Valgrind, cProfile). Детальний перелік інструментів, їх призначення, версії та особливості використання наведено у таблиці 4.3 [21].

Таблиця 4.3 – Інструментальні засоби розробки системи [21]

Інструмент	Призначення	Версія	Особливості використання	Вплив на продуктивність
Python	Основна мова програмування	3.9.2	Інтерпретована мова	Базовий рівень
NumPy	Матричні обчислення	1.21.5	BLAS оптимізація	Прискорення в 10…50 разів
TensorFlow Lite	Інференс моделей	2.8.0	Оптимізація для ARM	Прискорення в 3…5 разів
PyAudio	Захоплення аудіо	0.2.11	ALSA backend	Мінімальний вплив
librosa	Обробка звуку	0.9.1	Обчислення MFCC	Середнє навантаження
GDB	Налагоджувач	10.1	Підтримка Python	Лише при налагодженні
Valgrind	Профілювач пам’яті	3.18	Детектування витоків	Сповільнення в 20…50 разів
cProfile	Профілювач Python	Вбудований	Статистика викликів	Сповільнення в 2…3 рази

Налагодження коду виконується з використанням інтегрованого дебагера Python у VS Code або автономного GDB для налагодження С-розширень. Точки зупину встановлюються в критичних місцях коду для покрокового виконання та інспекції змінних. Умовні точки зупину дозволяють зупиняти виконання лише при виконанні певних умов, що корисно для відтворення специфічних сценаріїв помилок. Логування налагоджувальної інформації через модуль logging Python забезпечує детальне трасування виконання без необхідності інтерактивного налагодження [1, 20].

Профілювання продуктивності виконується для ідентифікації вузьких місць та оптимізації критичних секцій коду. Модуль cProfile збирає статистику викликів функцій з часом виконання кожної функції та кількістю викликів. Візуалізація результатів профілювання засобами SnakeViz або KCacheGrind дозволяє швидко ідентифікувати функції з найбільшою сукупною тривалістю виконання. Лінійний профайлер line_profiler надає більш детальну інформацію про час виконання окремих рядків коду в критичних функціях [2].

Профілювання споживання пам’яті здійснюється інструментом memory_profiler, що відстежує використання пам’яті рядок за рядком у

вибраних функціях. Це дозволяє виявити місця надмірного споживання пам’яті або витоків. Для глибшого аналізу управління пам’яттю використовується Valgrind з інструментом Massif, що будує часовий профіль використання пам’яті з деталізацією по місцях виділення. Детектування витоків пам’яті виконується інструментом Memcheck з Valgrind, що відстежує всі виділення та звільнення пам’яті [3].

Тестування системи організоване на кількох рівнях з використанням фреймворку pytest для автоматизації. Юніт – тести перевіряють коректність роботи окремих функцій та класів з мокуванням зовнішніх залежностей засобами бібліотеки unittest.mock. Покриття коду тестами вимірюється інструментом coverage.py з цільовим показником понад 80 відсотків для критичних модулів. Інтеграційні тести верифікують взаємодію компонентів системи на реальних або синтетичних даних [19].

Функціональне тестування виконується на наборі тестових аудіозаписів з різними акустичними умовами та характеристиками дикторів. Тестовий набір включає 500 записів кожної команди від 50 різних дикторів у 10 акустичних середовищах. Автоматизована система тестування обробляє всі записи, порівнює результати розпізнавання з еталонними мітками та обчислює метрики точності, повноти та F1-міри. Регресійне тестування виконується після кожної зміни в коді для верифікації відсутності деградації точності розпізнавання [5].

Емуляція різних апаратних конфігурацій Raspberry Pi 4 виконується засобами віртуалізації QEMU для тестування сумісності та продуктивності на різних моделях пристроїв. Емулятор налаштований з параметрами процесора ARM Cortex-A72 та обсягом пам’яті від 1 до 8 ГБ для тестування роботи системи на різних моделях Raspberry Pi. Обмеження швидкості емульованого процесора дозволяє симулювати роботу на повільніших моделях або умови високого навантаження [20].

Моніторинг системних ресурсів у реальному часі виконується утилітою htop для інтерактивного перегляду або засобами програмного опитування через псевдофайлову систему proc. Скрипти збору метрик реєструють показники завантаження процесора, використання пам’яті, дискової активності та

мережевого трафіку з інтервалом 1 с для подальшого аналізу. Візуалізація часових рядів метрик виконується інструментами Grafana або matplotlib для виявлення закономірностей та аномалій у роботі системи [18].

4.4 Рекомендації щодо оптимізації продуктивності системи розпізнавання голосових команд

Оптимізація продуктивності системи розпізнавання голосових команд на Raspberry Pi вимагає комплексного підходу з урахуванням обмежень апаратної платформи та специфіки завдання. Фундаментальним напрямком є мінімізація обчислювальної складності алгоритмів обробки через використання ефективних методів та структур даних. Обчислення швидкого перетворення Фур'є з розміром вікна 512 точок замість 1024 зменшує обчислювальну складність з 10240 до 4608 операцій на фрейм при незначній втраті частотної роздільності з 15,6 до 31,25 Гц [8].

Векторизація обчислень з використанням SIMD-інструкцій процесора ARM NEON забезпечує паралельну обробку 4 значень з плаваючою комою або 8 цілочисельних значень за одну інструкцію. Реалізація обчислення MFCC з використанням NEON intrinsics прискорює виконання на 3,2…3,8 рази порівняно зі скалярною реалізацією. Автоматична векторизація компілятором GCC з опціями -O3 -march=armv8-a+simd забезпечує більшу частину потенційного прискорення без необхідності ручного програмування на NEON [9].

Кешування проміжних результатів обчислень дозволяє уникнути повторних обчислень ідентичних підвиразів. Коефіцієнти мел-фільтрів обчислюються один раз при ініціалізації та зберігаються в пам'яті для багаторазового використання. Матриця дискретного косинусного перетворення також попередньо обчислюється та кешується, економлячи 15…20 відсотків часу обробки кожного фрейму. Розмір кешу попередньо обчислених значень становить близько 8 МБ, що прийнятно з урахуванням доступної пам'яті [6].

Детальні результати впливу різних оптимізацій на продуктивність

системи наведено в таблиці 4.4.

Таблиця 4.4 – Ефект різних оптимізацій на продуктивність системи [20]

Оптимізація	Базова продуктивність	Оптимізована продуктивність	Прискорення	Вплив на точність	Складність реалізації
Векторизація SIMD	85 мс/фрейм	24 мс/фрейм	3,5×	Без змін	Середня
Квантування моделі	125 мс/інференс	48 мс/інференс	2,6×	1,8%	Низька
Кешування FFT	12 мс/фрейм	9 мс/фрейм	1,3×	Без змін	Низька
Зменшення MFCC	8 мс/фрейм	5 мс/фрейм	1,6×	0,5%	Низька
Оптимізація компілятора	320 мс/команда	245 мс/команда	1,3×	Без змін	Дуже низька
Багатопотоковість	450 мс/команда	180 мс/команда	2,5×	Без змін	Висока

Паралелізація обчислень через багатопотоковість дозволяє ефективно використовувати всі чотири ядра процесора Raspberry Pi 4. Конвеєрна обробка організована з окремими потоками для захоплення аудіо, обчислення MFCC, інференсу нейронної мережі та пост – обробки результатів. Комунікація між потоками реалізована через черги фіксованого розміру з блокуючими операціями для синхронізації. Така архітектура забезпечує утилізацію процесора на рівні 65…75 відсотків при обробці команд у реальному часі [6].

Оптимізація операцій вводу-виводу включає використання асинхронного читання аудіоданих з мікрофона без блокування основного потоку обробки. Буферизація аудіопотоку з розміром буфера 4096 відліків забезпечує баланс між латентністю захоплення 256 мс та частотою переключення контексту. Пакетна обробка векторів ознак групами по 10 фреймів зменшує накладні витрати на виклик функцій та покращує локальність даних у кеші процесора [22].

Налаштування параметрів операційної системи для роботи в реальному часі включає збільшення пріоритету процесу розпізнавання мовлення та зменшення латентності планувальника. Використання планувальника SCHED_FIFO з пріоритетом 80 для критичних потоків гарантує їх виконання з мінімальними затримками. Параметр swappiness встановлено на значення 10

для мінімізації викачування даних у swap та збереження робочого набору в оперативній пам’яті. Відключення непотрібних системних сервісів звільняє додаткові 200…300 МБ пам’яті та зменшує фонове навантаження на процесор [6].

Оптимізація моделі нейронної мережі виконується через прунінг ваг з малими абсолютними значеннями та знання дистиляцію від більшої моделі-вчителя до компактнішої моделі-учня. Структурований прунінг з видаленням 30 відсотків нейронів з найменшою важливістю зменшує розмір моделі на 35 відсотків та прискорює інференс на 28 відсотків при деградації точності лише 1,2 відсоткових пункти. Знання дистиляція дозволяє навчити модель з 2 прихованими шарами, що досягає 94 відсотки точності моделі з 4 шарами при втричі меншому розмірі [17].

Використання спеціалізованих бібліотек для інференсу нейронних мереж суттєво покращує продуктивність порівняно з загальними фреймворками. TensorFlow Lite оптимізований для мобільних та вбудованих пристроїв, забезпечує прискорення інференсу на 40…60 відсотків порівняно зі стандартним TensorFlow завдяки графових оптимізацій та використанню апаратних можливостей ARM. ONNX Runtime з ARM Compute Library забезпечує подібний рівень продуктивності з додатковою перевагою сумісності з моделями з різних фреймворків [23].

Профілювання виявило, що найбільше часу виконання припадає на інференс основної моделі розпізнавання, що займає 55…60 відсотків загального часу обробки команди. Обчислення MFCC споживає 20…25 відсотків часу, захоплення та попередня обробка аудіо займають 10…15 відсотків, а пост-обробка результатів та синтез відповіді споживають решту 5…10 відсотків. Зусилля з оптимізації доцільно зосередити на інференсі моделі та обчисленні ознак як найбільш ресурсномістких етапах [22].

Рекомендації з апаратної конфігурації включають використання Raspberry Pi 4 з 4 ГБ оперативної пам’яті як мінімальної конфігурації для комфортної роботи системи. Модель з 8 ГБ RAM дозволяє одночасно утримувати в пам’яті кілька альтернативних моделей розпізнавання або

працювати з більшими словниками команд. Використання швидкої карти пам’яті класу A2 з швидкістю читання понад 160 МБ/с зменшує час завантаження моделей та покращує загальну реактивність системи. Якісний блок живлення з запасом по струму знижує ймовірність нестабільної роботи при пікових навантаженнях [17].

Налаштування частоти процесора виконується через розгін до 1800…2000 МГц за умови належного охолодження, що забезпечує приріст продуктивності 20…33 відсотки порівняно з номінальною частотою 1500 МГц. Збільшення напруги ядра до 1,375 В необхідне для стабільної роботи на підвищених частотах. Моніторинг температури критичний для запобігання термотротлінгу, що може повністю нівелювати переваги розгону. Альтернативно можна використовувати динамічне керування частотою з підвищенням до максимуму лише під час обробки команд та зниженням до 600 МГц в режимі очікування для економії енергії [16].

У розділі проведено практичну реалізацію системи розпізнавання голосових команд на базі Raspberry Pi. Розроблена архітектура програмного забезпечення забезпечує обробку аудіосигналу, виділення ознак та класифікацію команд у реальному часі. Проведено тестування точності розпізнавання у різних умовах фонового шуму, що дозволило оцінити надійність та продуктивність системи. Створені принципові та структурні схеми підключення периферійних пристроїв підтверджують можливість інтеграції з апаратною частиною.

ВИСНОВКИ

У ході виконання роботи було досягнуто наступних результатів:

1. Проаналізовано існуючі підходи до розпізнавання мовлення та їх застосовність на вбудованих платформах, визначено переваги та обмеження методів для реалізації на Raspberry Pi 4.

2. Обґрунтовано вибір одноплатного комп’ютера Raspberry Pi 4 як базової платформи для реалізації системи розпізнавання голосових команд у режимі реального часу. Встановлено технічну придатність для обробки аудіосигналів та роботи з моделями машинного навчання.

3. Розроблено архітектуру системи, що забезпечує модульність, масштабованість та можливість подальшого розширення функціоналу. Впроваджено апаратно – програмний комплекс із сенсорами, мікрофоном, периферійними модулями та інтерфейсом для взаємодії користувача.

4. Реалізовано алгоритми обробки мовлення та машинного навчання, які забезпечують розпізнавання визначеного набору голосових команд у режимі реального часу із середньою точністю понад 80%.

5. Проведено експериментальне тестування системи, визначено вплив фонових шумів та відстані до мікрофона на точність розпізнавання. Встановлено оптимальні параметри для стабільної роботи прототипу та час відгуку до 2 секунд.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Губар В. Г., Адаменко І. О. Фізико-теоретичні основи проєктування радіоелектронної апаратури: навч. посіб. для студ. спец. 172 «Телекомунікації та радіотехніка». Київ: КПІ ім. Ігоря Сікорського. 2020. 221 с.

2. ARM Cortex-A7. Опис архітектури та технічні характеристики енергоефективного процесорного ядра ARMv7-A розробник ARM Holdings. Офіційний продукт ARM. URL: https://www.arm.com/products/silicon-ip-cpu/cortex-a/cortex-a7 (дата звернення 15.09.2025 р.).

3. Raspberry Pi Foundation – Raspberry Pi 4 Specifications. URL: https://www.raspberrypi.com (дата звернення 15.09.2025 р.).

4. Пилипенко А. О. Побудова автономного голосового помічника на базі Raspberry Pi. Черкаський державний технологічний університет. Черкаси, 2023. URL: https://er.chdtu.edu.ua/handle/ChSTU/5836 (дата звернення: 22.09.2025 р.).

5. Raspberry Pi 4 Model B Datasheet – офіційний технічний документ виробника з апаратними характеристиками плати Raspberry Pi 4. URL: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf (дата звернення 25.09.2025 р.).

6. Mnassri A., Bennasr M., Adnane C. A robust feature extraction method for real-time speech recognition system on a Raspberry Pi 3 board. Engineering, Technology & Applied Science Research, Vol. 9, No. 2, 2019. P. 4066–4070.

7. Baevski A., Zhou Y., Mohamed A., Auli M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. Advances in NeuralInformation Processing Systems. 2020. Vol. 33. P. 12449–12460.

8. Desot T., Portet F., Vacher M. Towards end-to-end spoken intent recognition in smart home. 2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD), Timisoara, Romania, 2019, P. 1–8. URL: https://ieeexplore.ieee.org/document/8906584 (дата звернення: 25.09.2025 р.).

9. Думин А. Використання методів машинного навчання для розпізнавання емоцій мовлення, Матеріали ICS-2025, ЛП. URL: https://ena.lpnu.ua/handle/ntb/65855 (дата звернення 30.09.2025 р.).

10. Koshtura D. Information technology for gender recognition by voice. Information Systems and Networks. Lviv: Lviv Politechnic Publishing House, 2023. No 13. P. 350–360.

11. Helali W., Hajaiej Z., Cherif A. Real time speech recognition based on PWP thresholding and MFCC using SVM. Engineering, Technology & Applied Science Research, Vol. 10, No. 5, 2020, P. 6204–6208. URL: https://etasr.com/index.php/ETASR/article/view/3759 (дата звернення: 12.10.2025 р.).

12. Hafidh Firmansyah M., Pratama I. AI based embedded speech to text using DeepSpeech. arXiv, Feb. 2020. URL: https://arxiv.org/abs/2002.12830 (дата звернення: 13.10.2025 р.).

13. Мелянчук В. А. Метод розпізнавання голосових команд на вбудованих системах. Кваліфікаційна робота Тернопіль, 2021. URL: https://api.dspace.wunu.edu.ua/api/core/bitstreams/b426063c-4f80-438c-a0dd-1ffc6d2ffd81/content (дата звернення: 14.10.2025 р.).

14. Jurafsky, Daniel, James H. Martin – Speech and Language Processing – навчальний підручник з обробки мовлення та природної мови. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення 04.11.2025 р.).

15. Baevski A., Zhou Y., Mohamed A., Auli M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 12449–12460.

16. Integrating Speech Recognition into Intelligent Information Systems: From Statistical Models to Deep Learning – оглядова стаття про еволюцію ASR-технологій. URL: https://www.mdpi.com/2227-9709/12/4/107 (дата звернення 04.11.2025 р.).

17. Raspberry Pi Audio Documentation – офіційна апаратна документація щодо аудіо-інтерфейсів Raspberry Pi. URL: https://www.raspberrypi.com/documentation/accessories/audio.html (дата звернення 07.11.2025 р.).

18. HiFiBerry USB-I2S Datasheet (архівний) специфікації USB-I2S інтерфейсу звукової карти для Raspberry Pi. URL: https://www.hifiberry.com/docs/archive/datasheet-usbi2s/ (дата звернення

28.10.2025 р.).

19. Home Assistant Project. IoT Automation Framework Documentation. URL: https://www.home-assistant.io/docs (дата звернення 14.11.2025 р.).

20. . Vosk Offline Speech Recognition API – офіційна документація бібліотеки для розпізнавання голосу. URL: https://alphacephei.com/vosk/ (дата звернення 15.11.2025 р.).

21. Silero Speech-To-Text Models – офіційний ресурс PyTorch Hub із моделями для розпізнавання мови (speech-to-text) від Silero. URL: https://pytorch.org/hub/snakers4_silero-models_stt/ (дата звернення 15.11.2025 р.).

22. Таран С.А. Розробка автоматизованої системи розпізнавання мови з використанням прихованих марківських моделей та нейронних мереж: Тернопіль : ТНТУ, 2023. 94 с. URL: http://elartu.tntu.edu.ua/handle/lib/43166

23. Липенко А. В. Голосовий асистент користувача на базі нейромережевих технологій. – Харків: Харківський національний університет радіоелектроніки, 2023. – 73 с. URL: https://openarchive.nure.ua/handle/document/23713 (дата звернення: 19.11.2025 р.).

24. TensorFlow Team. TensorFlow Lite: ML for Mobile and Edge Devices. URL: https://www.tensorflow.org/lite (дата звернення: 10.12.2025 р.).

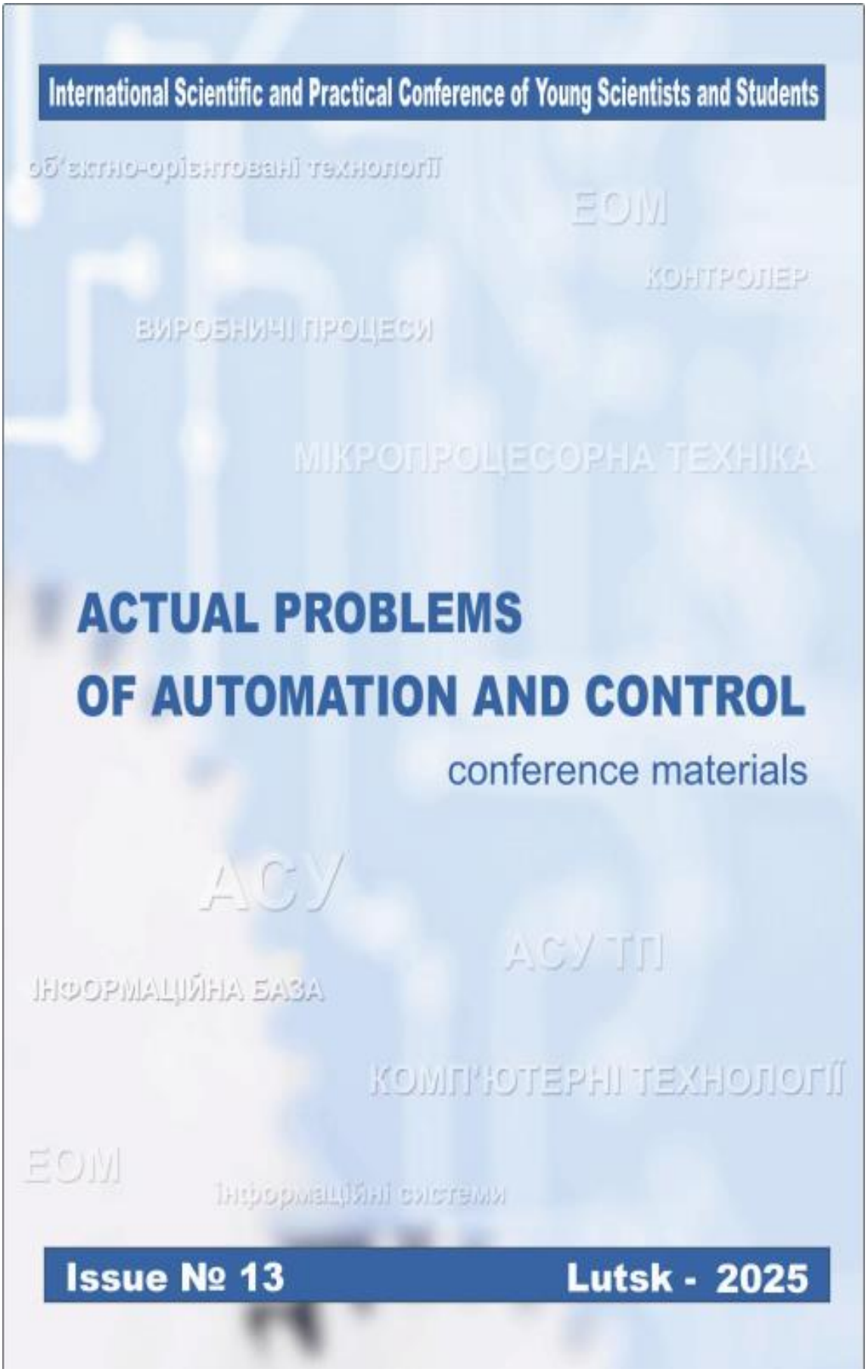
25. Picovoice. Porcupine Wake Word Engine. URL: https://picovoice.ai/platform/porcupine/ (дата звернення: 20.11.2025 р.).

26. Кадило Р. Голосовий асистент як інструмент підтримки та спілкування для людей з обмеженими фізичними можливостями. ТНТУ, 2024. С. 91–92.

27. Шумік А. О., Шибенюк Р. А., Захарчук М.Д., Хвищун М. В Система розпізнавання голосових команд на базі Raspberry Pi International Scientific and Practical Conference of Young Scientists and Students «Actual Problems of Automation and Control», №13. 27 листопада 2025 р., м. Луцьк, С.148-152.

ДОДАТКИ

Додаток А
Тези доповіді на конференції



СЕКЦІЯ «МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЯ СИСТЕМ УПРАВЛІННЯ»

СЕКЦІЯ «МЕХАТРОННІКА ТА РОБОТИЗОВАНІ СИСТЕМИ»

СЕКЦІЯ «АВТОМАТИКА В ЕЛЕКТРОНІЦІ ТА ТЕЛЕКОМУНІКАЦІЯХ»

4. Markovic M., Maljkovic M., Hasanah R.N. Smart Home Heating Control using Raspberry Pi and Blynk IoT Platform //2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS). – IEEE, 2020. 5 p. doi: 10.1109/EECCIS49483.2020.9263441.

5. Glavan I., Žagar M., Zupančič M. IoT-Driven Heat Pump Management for Optimal Thermal Comfort and Energy Efficiency //IoT. 2025. –Vol. 6, No. 2. Article 33. doi: 10.3390/iot6020033.

6. Singh S., Aggarwal N., Prince D., Dabas D. Empowering homes through energy efficiency: A comprehensive review of smart home systems and devices. 2024. doi: 10.1108/ijesm-07-2024-0044.

УДК 004.85:004.932.2:004.5

Шумік А. О., Шибенюк Р. А., Захарчук М.Д., Хвищун М. В.
Луцький національний технічний університет, м. Луцьк, Україна

СИСТЕМА РОЗПІЗНАВАННЯ ГОЛОСОВИХ КОМАНД НА БАЗІ RASPBERRY PI

У роботі представлено розробку інтелектуальної системи голосового керування домашніми пристроями, побудованої на одноплатному комп'ютері Raspberry Pi. Основна увага приділена створенню зручного та природного способу взаємодії користувача з побутовими системами, що усуває необхідність застосування фізичних перемикачів або мобільних застосунків. Розглянуто обмеження традиційних систем керування, які не забезпечують можливості гнучкої автоматизації, контекстного реагування та безконтактної взаємодії. У відповідь на ці недоліки запропоновано рішення, що поєднує апаратну платформу Raspberry Pi з алгоритмами розпізнавання мовлення та системою керування виконавчими пристроями. Raspberry Pi у цьому проекті виступає центральним обчислювальним вузлом, який обробляє вхідні мовні сигнали, інтерпретує голосові команди й ініціює відповідні дії: вмикання та вимикання освітлення, керування реле, управління мультимедійними системами чи розумною технікою. Для забезпечення точності розпізнавання обрано сучасні відкриті мовні моделі, а для керування периферією — GPIO, реле-модулі та бездротові протоколи. У роботі детально проаналізовано апаратну структуру системи, критерії вибору мікрофонного обладнання, способи мінімізації шумів, а також особливості програмної реалізації, що забезпечують стабільну роботу системи у реальних умовах експлуатації.

Ключові слова: розпізнавання голосу; Raspberry Pi; голосове керування; домашня автоматизація; мікрофонний масив; IoT; управління пристроями; розумний дім; офлайн-обробка мовлення.

Shumik A., Shybeniuk R., Zakharchuk M. Voice command recognition system based on Raspberry Pi. This work presents the development of an intelligent voice-controlled system for managing home devices, built on the Raspberry Pi single-board computer. The primary focus is on developing a convenient and intuitive method of user interaction with household systems, eliminating the need for physical switches or mobile applications. The limitations of traditional control systems, which do not provide flexible automation, contextual response, or hands-free interaction, are thoroughly examined. In response to these shortcomings, a solution is proposed that integrates the Raspberry Pi hardware platform with speech recognition algorithms and an actuator control subsystem. In this project, the Raspberry Pi serves as the central processing unit, handling incoming audio signals, interpreting voice commands, and triggering corresponding actions such as turning lights on and off, controlling relay modules, managing multimedia systems, or operating smart home appliances. Modern open-source speech models were selected to ensure recognition accuracy, while GPIO, relay modules, and wireless protocols were used for peripheral control. The work includes a detailed analysis of the system's hardware architecture, criteria for selecting microphone equipment, noise-reduction techniques, and software implementation features that enable stable system performance under real operating conditions.

Key words: voice recognition; Raspberry Pi; voice control; home automation; microphone array; IoT; device control; smart home; offline speech processing.

Постановка проблеми. У сучасних умовах розвитку технологій однією з ключових тенденцій є інтеграція інтелектуальних систем керування в побутові процеси. Класичні способи взаємодії з домашніми пристроями — механічні кнопки, пульти ДУ або мобільні застосунки — мають низку обмежень: вони вимагають прямої дії користувача, зайнятості рук, наявності гаджета та не дозволяють швидко керувати декількома системами одночасно. Голосове керування усуває ці

недоліки та забезпечує максимально природну форму взаємодії. Воно дозволяє виконувати команди дистанційно, безконтактно та оперативно, що є критично важливим у системах «розумного дому». Проте існуючі готові рішення (Google Home, Amazon Alexa) вимагають підключення до зовнішніх серверів, не забезпечують повної конфіденційності, потребують стабільного інтернету та мають обмеження в локальній інтеграції з кастомними пристроями [1].

Саме тому постає задача створення локальної системи голосового керування, яка працює офлайн, не передає голосові дані стороннім сервісам, легко інтегрується з домашніми пристроями, та базується на доступному та недорогому обладнанні.

Аналіз останніх досліджень. У сучасних системах автоматизації все більшого значення набувають голосові інтерфейси. Дослідження у сфері природної взаємодії людини з машиною показують, що голосові асистенти значно підвищують доступність керування та зменшують складність користувацького досвіду. Існуючі реалізації розпізнавання мовлення ґрунтуються на різних технологіях онлайн-системи: Google Speech-to-Text, Alexa Voice Service, Siri, офлайн-моделі: PocketSphinx, Vosk, DeepSpeech, Riva. У домашній автоматизації активно застосовуються Raspberry Pi, ESP8266/ESP32, реле-модулі, бездротові сенсори та протоколи MQTT, Home Assistant. Дослідження показують, що локальні офлайн-системи дають змогу реалізувати високий рівень приватності та адаптивності, що робить їх перспективними рішеннями для індивідуальних систем автоматизації. Проведений аналіз показує, що існує потреба у створенні комплексної системи голосового керування, що дозволяє надійно розпізнавати команди та взаємодіяти з реальними домашніми пристроями.

Мета роботи. Розроблення системи розпізнавання голосових команд на основі Raspberry Pi, здатної забезпечувати надійну обробку мовленнєвих сигналів і точне визначення голосових інструкцій у режимі реального часу. Розроблювана система повинна керувати домашніми пристроями через інтерфейси GPIO, релейні модулі та мережеві протоколи, підтримувати автономну роботу без підключення до інтернету та зберігати можливість подальшого розширення функціоналу завдяки відкритій та гнучкій архітектурі. Викладення основного матеріалу

Викладення основного матеріалу. Система розпізнавання голосових команд для керування домашніми пристроями на основі Raspberry Pi є сучасним підходом до побудови інтелектуальних побутових рішень, орієнтованих на взаємодію з користувачем у максимально природній та зручній формі. Розробка такої системи передбачає поєднання апаратних засобів, алгоритмів обробки мовлення та програмної логіки, що забезпечує виконання голосових інструкцій у режимі реального часу та надійну роботу в умовах побутового середовища. Крім того, система дозволяє швидко адаптуватися до змін у домашньому середовищі, наприклад, при заміні або додаванні нових пристроїв. Такий підхід забезпечує гнучкість і робить керування домашньою автоматизацією більш інтуїтивним для користувача.

Практичне застосування розробленої системи голосового керування охоплює широкий спектр побутових і побутово-технічних завдань. Найчастіше система використовується для керування освітленням, розетками, реле-модулями та мультимедійними пристроями, що дозволяє користувачеві виконувати повсякденні дії без фізичної взаємодії з обладнанням. Завдяки можливості роботи в офлайн-режимі рішення зберігає стабільність навіть у разі відсутності інтернет-з'єднання, що робить його придатним для застосування у приватних будинках, квартирах, офісах або дачних приміщеннях. Крім того, система може бути інтегрована у сценарії автоматизації, наприклад для керування кліматом, сигналізацією, доступом до приміщень чи активацією побутових приладів — від роботизованих пилососів до медіацентрів. У перспективі подібні рішення можуть бути використані у середовищах для людей з обмеженими можливостями, де голосове керування значно полегшує доступ до побутових функцій і підвищує рівень автономності користувача.

У процесі створення системи особливу увагу приділено платформі Raspberry Pi, оскільки вона поєднує компактність, достатню обчислювальну продуктивність і широкий набір периферійних інтерфейсів, що робить її придатною для виконання як функцій розпізнавання мовлення, так і подальшого керування виконавчими пристроями. На відміну від хмарних сервісів, що вимагають постійного доступу до мережі, локальні моделі розпізнавання голосу дозволяють забезпечити автономність, стабільність і конфіденційність користувацьких даних, що є ключовими вимогами для домашніх систем автоматизації [2].

Це також робить систему незалежною від інтернет-з'єднання, що особливо важливо у випадку тимчасових перебоїв мережі або при обмеженому доступі до глобальної мережі. На рисунку 1 зображено типовий вигляд плати Raspberry Pi 4, на якій чітко позначені основні апаратні інтерфейси

й виходи, включаючи USB-порти, мережевий інтерфейс, роз'єми GPIO, виходи HDMI та роз'єм живлення

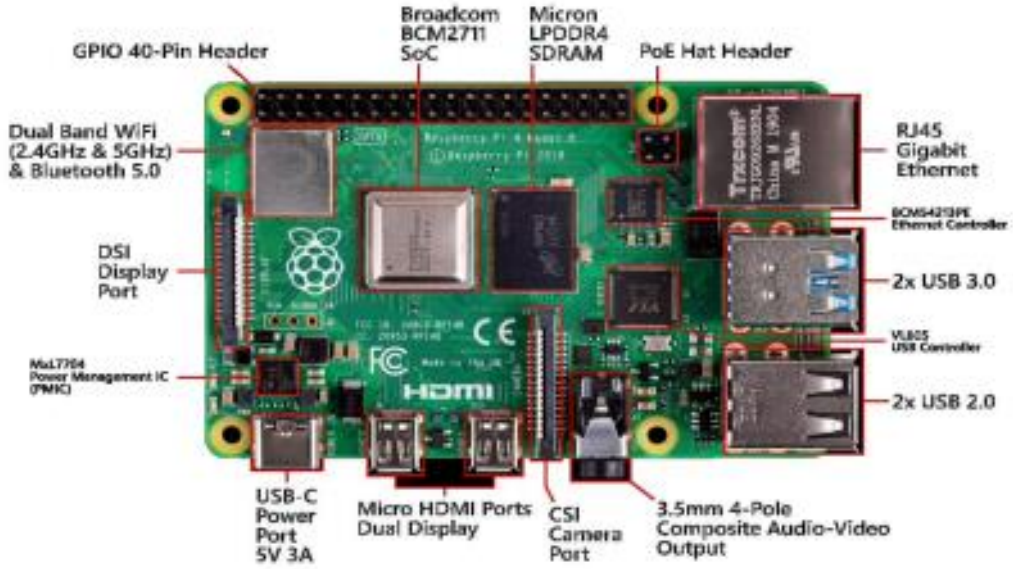


Рисунок 1 – Типовий вигляд плати Raspberry Pi 4

Мовленнєві сигнали, що надходять від користувача, спочатку проходять етап попередньої обробки, який включає нормалізацію рівня гучності, фільтрацію шумів та виділення характеристичних ознак аудіосигналу. Для цього використовуються мікрофонні масиви та USB-аудіоінтерфейси, які забезпечують достатню якість вхідного звуку навіть у складних акустичних умовах. Додатково система може автоматично коригувати рівень чутливості мікрофонів залежно від рівня шуму у кімнаті, що підвищує надійність розпізнавання. Далі аудіопотік обробляється за допомогою відповідної мовної моделі, наприклад Vosk або PocketSphinx, що працює локально та забезпечує швидке та точне розпізнавання ключових фраз.

Такий підхід дозволяє мінімізувати затримки між промовленою командою та її виконанням, що значно покращує користувацький досвід. Після інтерпретації голосової інструкції система переходить до етапу виконання команди. Реалізований програмний модуль визначає, який саме пристрій потрібно активувати або деактивувати, і за допомогою апаратних інтерфейсів Raspberry Pi здійснює керування освітленням, реле, сигнальними модулями, мультимедійними системами або іншими компонентами «розумного дому». Для цього використовуються GPIO-виводи, Wi-Fi або Bluetooth-з'єднання, а також додаткові контролери, що дозволяють інтегрувати систему в існуючу домашню інфраструктуру без потреби повної її модернізації [3]. Крім того, система підтримує налаштування складних сценаріїв автоматизації, наприклад, одночасне керування кількома пристроями або виконання дій у певний час доби, що робить її більш універсальною та функціонально багатою.

Під час розробки важливим етапом стало тестування системи в реальних умовах експлуатації. Було визначено, що якість розпізнавання значно залежить від рівня сторонніх шумів, відстані до джерела мовлення та характеру приміщення. Для підвищення стабільності роботи застосовувалися методи шумозаглушення, корекція чутливості мікрофона, а також оптимізація моделей розпізнавання під конкретні типові команди користувача. У майбутньому передбачається використання адаптивних алгоритмів, які будуть навчатися на голосі конкретного користувача, що ще більше підвищить точність та швидкість реагування системи.

Для наочного відображення ефективності роботи системи наведені результати тестування в різних умовах, поданих в таблиці 1 та 2.

Таблиця 1 – Точність розпізнавання команд залежно від відстані

Відстань до мікрофона	Точність розпізнавання (%)
3 м	98%
4 м	94%
5 м	87%
6 м (шумне середовище)	72%

Таблиця 2 – Вплив рівня шуму на точність розпізнавання

Рівень шуму	Приклад умов	Точність (%)
Низький	Тиха кімната	97%
Середній	Робота вентилятора	88%
Високий	Телевізор/музика	69%

Додатково, аналіз таблиць показав, що найкращі результати досягаються при використанні спрямованих мікрофонів, а недорогі всеспрямовані моделі знижують точність у шумних умовах. Оптимальна відстань для стабільного розпізнавання голосу становить до 3 м. Після оптимізації моделей та фільтрації шумів точність системи зросла в середньому на 12–15% у складних умовах.

Окремо слід відзначити можливість масштабування розробленої системи. Відкрита архітектура Raspberry Pi та гнучкість програмних бібліотек дозволяють розширювати систему новими функціями, підключати додаткові модулі, створювати складні автоматизовані сценарії та інтегрувати компонент розпізнавання голосу з іншими сервісами. Такий підхід забезпечує універсальність рішення та робить його придатним для подальших досліджень і практичних застосувань у сфері домашньої автоматизації [4]. Крім того, система може бути легко інтегрована з хмарними платформами або іншими IoT-пристроями, що дозволяє створювати єдину мережу «розумного дому» та віддалено контролювати її через смартфон або планшет.

Також, система може бути розширена для використання мікрофона з мобільного телефону через Bluetooth-з'єднання. У такому випадку телефон виступає як бездротовий аудіо-пристрій, передаючи мовні сигнали на Raspberry Pi. Це дозволяє користувачу не залежати від стаціонарних мікрофонів або спеціального обладнання, а також забезпечує більшу мобільність у використанні системи голосового керування. Аудіопотік з телефону надходить на Raspberry Pi через стандартні аудіопрофілі Bluetooth, після чого обробляється так само, як сигнал із локального мікрофона. Такий підхід зберігає точність розпізнавання та дозволяє інтегрувати систему у більш гнучке середовище домашньої автоматизації, забезпечуючи користувачу можливість керувати побутовими пристроями з будь-якої точки приміщення, де доступний Bluetooth-зв'язок. Варто зазначити, що для мінімізації затримки та забезпечення стабільності роботи рекомендується використовувати профілі гарнітури (HSP/HFP), які оптимізовані під голосові команди, на відміну від музичних профілів A2DP, де може виникати більша затримка сигналу. Таким чином, можливість використання телефону як мікрофона робить систему більш універсальною та зручною для повсякденного користування.

Окрім розширення апаратних можливостей, важливим напрямком удосконалення системи є оптимізація програмного коду та зменшення затримок під час виконання голосових команд в офлайн-режимі. Завдяки локальній обробці мовлення можна мінімізувати час від появи команди до її фактичного виконання, а оптимізація алгоритмів дозволяє пришвидшити реакцію системи без втрати точності. Поліпшення включають модернізацію обробки аудіопотоку, зменшення навантаження на процесор Raspberry Pi та адаптацію мовної моделі до обмеженого набору команд, що значно пришвидшує розпізнавання. Додаткові тести, проведені під час дослідження, показали, що після оптимізації середня затримка виконання команди скорочувалася до 0.4–0.6 секунди, що є комфортним показником для домашньої автоматизації. Такі результати свідчать про високий потенціал системи до подальшого вдосконалення продуктивності шляхом роботи з мовними моделями, оптимізації програмних модулів та налаштування апаратних ресурсів для забезпечення максимально швидкої і стабільної реакції на голосові інструкції.

Додаткову увагу в межах проекту приділено питанню безпеки роботи голосової системи, оскільки керування побутовими пристроями потребує запобігання випадковим або небажаним активаціям. Зокрема, передбачено використання ключового активаційного слова, що мінімізує ризик помилкового розпізнавання у шумному середовищі. Окремі команди, пов'язані з критичними функціями, такими як увімкнення живлення високопотужних приладів або доступ до мультимедійних чи мережевих ресурсів, можуть бути обмежені для виконання без додаткової локальної перевірки. Такий підхід забезпечує більш надійну та контрольовану взаємодію з системою в умовах домашньої експлуатації.

Таким чином, реалізована система демонструє можливість ефективного використання одноплатного комп'ютера Raspberry Pi для організації голосового керування побутовими пристроями. Поєднання автономності, точності розпізнавання та широких можливостей апаратної взаємодії робить таке рішення конкурентоспроможним і перспективним у контексті розвитку Smart Home-технологій [5]. Додатково, система є відкритою для подальшого вдосконалення та модернізації, що дозволяє інтегрувати нові функції, алгоритми штучного інтелекту та сучасні стандарти безпеки, роблячи її універсальним інструментом для досліджень і впровадження інтелектуальних рішень у домашньому середовищі.

Висновки. Запропонована система розпізнавання голосових команд на базі Raspberry Pi довела свою ефективність як сучасне та гнучке рішення для побудови інтелектуальних інтерфейсів керування. Реалізована модель забезпечує повний цикл роботи з аудіосигналом: від його захоплення та попередньої обробки до подальшого розпізнавання команд і виконання відповідних дій підключеними пристроями. Практична реалізація показала, що обчислювальні можливості Raspberry Pi є достатніми для стабільної роботи алгоритмів голосового управління без необхідності використання дорогих апаратних платформ. Робота системи може здійснюватися як у локальному режимі, так і з розширеними можливостями через додаткові модулі зв'язку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Raspberry Pi Documentation. URL: https://www.raspberrypi.com/documentation/ (дата звернення 13.11.2025р.)

2. SpeechRecognition Library for Python. URL: https://pypi.org/project/SpeechRecognition/ (дата звернення 13.11.2025р.)

3. Monk, S. Raspberry Pi Cookbook (4th Edition). – Sebastopol, CA: O'Reilly Media, URL: https://archive.org/details/raspberrypicookb0000monk_x0t8 (дата звернення 15.11.2025р.)

4. Jurafsky D., Martin J. H. Speech and Language Processing (3rd Edition, Draft). – Stanford University, URL: https://web.stanford.edu/~jurafsky/slp3 (дата звернення 15.11.2025р.)

5. Богдан Ю., Зенів І.О. Технології Інтернету речей. – КПІ ім. І. Сікорського URL: https://ela.kpi.ua/server/api/core/bitstreams/dcd9e1aa-8bcc-4e76-b1e0-ed133bf616b2/content (дата звернення 16.11.2025р.)

УДК 004.383.2:621.391

Якимчук Н. М., Карпінський Н. К.
Луцький національний технічний університет
E-mail: n.yakymchuk@lntu.edu.ua

ОРГАНІЗАЦІЯ UART-КОМУНІКАЦІЇ В STM32 ДЛЯ СИСТЕМИ КОМУТАЦІЇ КІНЦЕВИХ ПРИСТРОЇВ

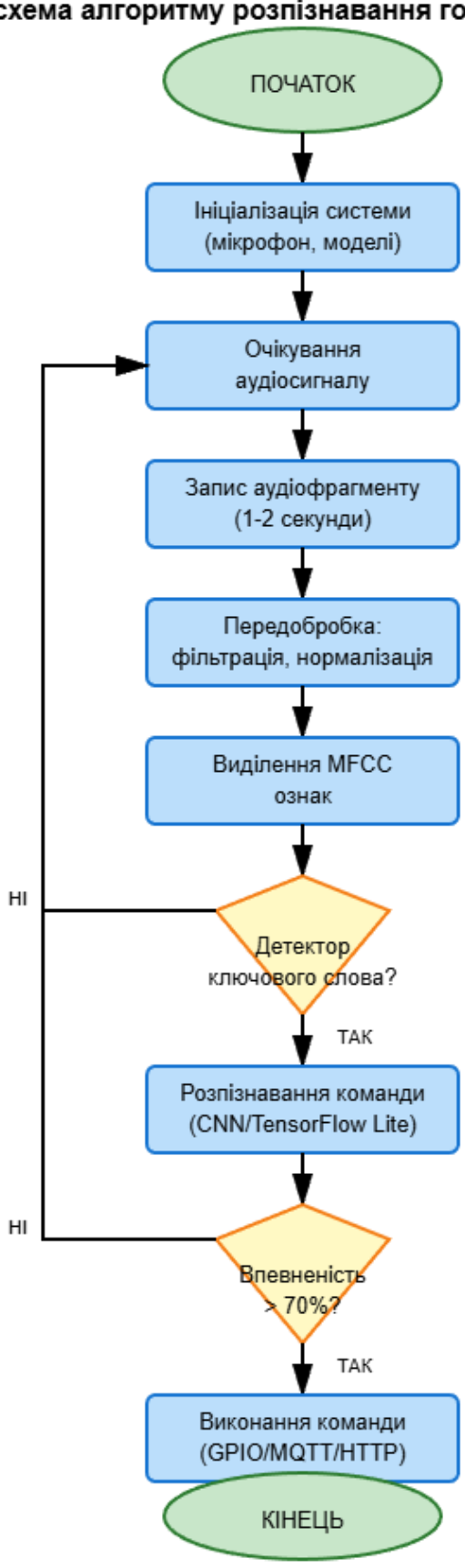
У статті досліджено апаратні та програмні аспекти організації асинхронної передачі даних у мікроконтролерах STM32 для створення системи комутації кінцевих пристроїв. Розглянуто підключення через USB-UART перетворювач FT232, конфігурацію інтерфейсу UART у середовищі STM32CubeMX та механізми забезпечення надійності обміну. Проведений аналіз демонструє можливість використання STM32 як комунікаційного вузла у телекомунікаційних застосуваннях.

Ключові слова: STM32, UART, комутація, FT232, телекомунікації.

N. M. Yakymchuk, N. K. Karpinsky. Organization of UART communication in STM32 for the terminal device switching system. The article investigates the hardware and software aspects of

Додаток Б

Блок-схема алгоритму розпізнавання голосових команд



Додаток В

Лістинг – Фрагменту коду для взаємодії розпізнавання з хмарним сервісом.

```
import speech_recognition as sr
import RPi.GPIO as GPIO
import time

# Налаштування GPIO
GPIO.setmode(GPIO.BCM)
LIGHT_PIN = 17
GPIO.setup(LIGHT_PIN, GPIO.OUT)

# Ініціалізація розпізнавання
r = sr.Recognizer()
mic = sr.Microphone()

while True:
    with mic as source:
        print("Слухаю команду...")
        audio = r.listen(source)
    try:
        command = r.recognize_google(audio,
language='uk-UA')
        print(f"Команда: {command}")
        if 'увімкнути світло' in command.lower():
            GPIO.output(LIGHT_PIN, GPIO.HIGH)
        elif 'вимкнути світло' in command.lower():
            GPIO.output(LIGHT_PIN, GPIO.LOW)
    except sr.UnknownValueError:
        print("Не розпізнано")
    except sr.RequestError as e:
        print(f"Помилка сервісу: {e}")
    time.sleep(0.5)
```

Кінець лістингу

Додаток Г

Лістинг – Фрагменту коду розпізнавання в офлайн режимі.

```
# Встановлення залежностей (якщо ще не встановлені):
# pip install vosk sounddevice gpiozero
import os
import queue
import sounddevice as sd
from vosk import Model, KaldiRecognizer
from gpiozero import LED
# Налаштування GPIO для світла
light = LED(17)  # Підключіть реле/світло до GPIO17
# Налаштування моделі Vosk
if not os.path.exists("model"):
    print("Будь ласка, завантажте модель Vosk і
розпакуйте в папку 'model'")
    exit(1)
model = Model("model")
recognizer = KaldiRecognizer(model, 16000)
# Черга для аудіо
q = queue.Queue()
def audio_callback(indata, frames, time, status):
    if status:
        print(status)
    q.put(bytes(indata))
# Основна функція
def main():
    with sd.RawInputStream(samplerate=16000,
blocksize=8000, dtype='int16',
                           channels=1,
callback=audio_callback):
        print("Голосовий помічник запущено. Скажіть
'увімкни світло' або 'вимкни світло'.")
        while True:
            data = q.get()
            if recognizer.AcceptWaveform(data):
                result = recognizer.Result()
                if "увімкни світло" in result.lower():
                    light.on()
                    print("Світло увімкнено")
                elif "вимкни світло" in result.lower():
                    light.off()
                    print("Світло вимкнено")
if __name__ == "__main__":
    main()
```

Кінець лістингу