УДК 004.415.2:004.934

Д.І. ЧЕКАЛЮК
Д.І. КАТЄЛЬНІКОВ

АРХІТЕКТУРА БАГАТОМОДУЛЬНОЇ СИСТЕМИ ГОЛОСОВОГО АСИСТЕНТА ДЛЯ АВТОМАТИЗАЦІЇ ОС НА БАЗІ ЕКОСИСТЕМИ PYTHON

Вінницький національний технічний університет

Анотація

Стаття присвячена проектуванню та розробці архітектури локального голосового асистента для автоматизації процесів операційної системи Windows. Здійснено детальний аналіз інтеграції різнорідних Python-бібліотек у єдиний асинхронний пайплайн. Описано механізми енергоефективного фонового прослуховування, транскрибації аудіо, семантичного аналізу тексту за допомогою великих мовних моделей, а також безпосередньої взаємодії з Windows API для виконання системних команд. Запропоновано рішення на основі динамічної архітектури плагінів.

Ключові слова: системне адміністрування; архітектура програмного забезпечення; динамічні плагіни.

Abstract

The article is devoted to the design and development of the architecture of a local voice assistant for the automation of Windows operating system processes. A detailed analysis of the integration of heterogeneous Python libraries into a unified asynchronous pipeline is presented. The mechanisms of energy-efficient background listening, audio transcription, semantic text analysis using large language models, and direct interaction with the Windows API for the execution of system commands are described. A solution based on a dynamic plugin architecture is proposed.

Keywords: system administration; software architecture; dynamic plugins; Python; Picovoice; OpenAI API; Edge TTS; SpeechRecognition; Windows API.

Постановка проблеми

Розвиток інтерфейсів взаємодії людини з комп'ютером (HCI) зміщує фокус у бік голосового керування. Проте створення ефективного голосового асистента, здатного не лише вести діалог, а й виконувати глибоку автоматизацію налаштувань операційної системи (ОС), є комплексним архітектурним завданням. Проблема полягає у необхідності об'єднання локальних низькорівневих інструментів (для роботи зі звуком та системними викликами) із високорівневими хмарними сервісами штучного інтелекту без втрати швидкодії та надмірного споживання апаратних ресурсів ПК. Метою дослідження є розробка оптимальної структури взаємодії спеціалізованих бібліотек мови Python [1] для реалізації такого комплексу.

Архітектура обробки аудіопотоку та Wake Word

Першим етапом роботи системи є постійне фонове очікування активації користувачем. Використання стандартних онлайн-рушіїв для цієї мети є неефективним через високе навантаження на мережу та порушення конфіденційності. Тому в якості підсистеми активації (Wake Word engine) інтегровано бібліотеку pvporcupine [2]. Вона забезпечує локальне розпізнавання ключових фраз з використанням глибоких нейронних мереж, мінімізуючи споживання ресурсів центрального процесора. Захоплення сирого аудіопотоку (PCM-даних) реалізовано через кросплатформну бібліотеку PyAudio, яка дозволяє гнучко налаштовувати розмір буфера (chunk size) та частоту дискретизації. Після спрацювання тригера аудіодані передаються до модуля SpeechRecognition, який відповідає за їхню онлайн-транскрибацію (Speech-to-Text)[3], перетворюючи голос у текстовий формат для подальшої обробки.

Семантичний аналіз та забезпечення відмовостійкості

Отриманий текст потребує інтелектуального аналізу для визначення намірів користувача (Intent

Recognition). Ця задача делегована бібліотеці openai [4], яка надає доступ до сучасних великих мовних моделей (LLM). Модель класифікує запит: формує текстову відповідь для користувача або генерує стандартизовану JSON-структуру для виклику конкретного системного плагіна. Оскільки мережева взаємодія з API є вразливою до таймаутів та збоїв з'єднання, в архітектуру інтегровано бібліотеку tenacity. Вона реалізує механізм повторних спроб (Retry mechanism) з експоненційною затримкою (exponential backoff), гарантуючи стабільну роботу комунікаційного рівня системи навіть за умов нестабільного інтернет-з'єднання.

Генерація мовлення (Text-to-Speech) та мультимедіа

На етапі синтезу мовлення було прийнято рішення відмовитися від важких локальних моделей, які потребують значних обчислювальних потужностей GPU. Замість цього імплементовано бібліотеку edge-tts, що взаємодіє з хмарними сервісами генерації мовлення. Це забезпечує високоякісне, природне звучання голосу асистента з мінімальною затримкою (latency). Відтворення згенерованих аудіофайлів здійснюється через мікшер бібліотеки pygame, що дозволяє паралельно обробляти декілька звукових потоків без блокування основного потоку виконання програми.

Інтеграція з ОС та механізм динамічних плагінів

Ключовою особливістю розробленого асистента є здатність керувати операційною системою. Для моніторингу стану апаратного забезпечення (завантаження CPU, використання RAM) використовується модуль psutil. Управління базовими налаштуваннями, такими як яскравість екрану, забезпечується бібліотекою screen-brightness-control. Більш складні задачі, наприклад, керування рівнем гучності окремих застосунків чи загальним аудіомікшером Windows, реалізовані через pycaw (Python Core Audio Windows Library). Для імітації дій користувача (натискання клавіш, комбінації) використовується бібліотека keyboard, а для прямої взаємодії з вікнами та процесами ОС — обгортка Windows API pywin32.

Усі ці функції інкапсульовані в окремі модулі за допомогою архітектури динамічних плагінів. Система автоматично сканує директорію плагінів під час запуску і реєструє доступні команди за допомогою механізмів рефлексії (reflection) мови Python, що дозволяє розширювати функціонал асистента без зміни ядра програми.

Фонова робота та розгортання

Зважаючи на специфіку голосового асистента, його графічний інтерфейс був мінімізований для забезпечення ненав'язливої роботи. За допомогою бібліотек pystray та Pillow реалізовано інтеграцію програми в системний трей (System Tray) Windows. Користувач отримує доступ до контекстного меню для управління станом асистента (пауза, вимкнення). Для розповсюдження продукту використовується утиліта pyinstaller, яка збирає інтерпретатор Python, усі вищезазначені залежності та ресурси в єдиний виконуваний файл (.exe), що робить програму автономною і готовою до запуску на кінцевих пристроях без додаткового налаштування середовища.

Висновки

Запропонована багатомодульна архітектура демонструє високу ефективність у вирішенні задач голосової автоматизації. Використання спеціалізованих бібліотек дозволило чітко розмежувати зони відповідальності: pvporcupine та PyAudio забезпечують надійне введення, OpenAI та edge-tts відповідають за інтелектуальну складову, а pywin32, pycaw та psutil надають глибокий доступ до ресурсів Windows[5]. Механізм динамічного завантаження плагінів гарантує високу масштабованість системи, а використання pyinstaller [6] та pystray забезпечує зручність кінцевого використання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The Python Standard Library: ctypes and dynamic loading [Electronic resource] / Python Software Foundation. – Electronic data. – 2025. – Mode of access: https://docs.python.org/3/library/ (date of access: 26.02.2026). – Title from the screen.
2. Porcupine Wake Word Engine Documentation [Electronic resource] / Picovoice Docs. – Electronic data. – 2025. – Mode of access: https://picovoice.ai/docs/porcupine/ (date of access: 26.02.2026). – Title from the screen.

3. Speech service documentation: Text-to-speech API [Electronic resource] / Microsoft Learn ; Azure AI Services. – Electronic data. – Mode of access: https://learn.microsoft.com/en-us/azure/ai-services/speech-service/ (date of access: 26.02.2026). – Title from the screen.
4. Chat Completions and Function Calling [Electronic resource] / OpenAI API Reference ; OpenAI Platform. – Electronic data. – Mode of access: https://platform.openai.com/docs/api-reference (date of access: 26.02.2026). – Title from the screen.
5. Python for Windows Extensions (pywin32) documentation [Electronic resource] / M. Hammond ; GitHub Pages. – Electronic data. – Mode of access: https://mhammond.github.io/pywin32/ (date of access: 26.02.2026). – Title from the screen.
6. PyInstaller Manual: Bundling Python applications [Electronic resource] / PyInstaller Development Team ; PyInstaller Docs. – Electronic data. – Mode of access: https://pyinstaller.org/en/stable/ (date of access: 26.02.2026). – Title from the screen.

Чекалюк Дмитро Ігорович – студент групи 4ПІ-22б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: ifoxps@gmail.com.

Катєльніков Денис Іванович. – к.т.н., доцент, доцент кафедри ПЗ, Вінницький національний технічний університет, м. Вінниця, e-mail: katielnikov@vntu.edu.ua.

Chekaliuk Dmytro Ihorovych – student of the 4PI-22b group, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: ifoxps@gmail.com.

Katielnikov Denis I. – Ph.D., Associate Professor, Department of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: katielnikov@vntu.edu.ua.