ВСТУП

У сучасному світі стрімкий розвиток технологій побутової автоматизації сприяє поширенню концепції «розумного будинку», де повсякденні пристрої стають не просто виконавцями окремих команд, а повноцінними учасниками інтегрованої системи. Одним із таких пристроїв є робот-пилосос — багатофункціональний помічник, здатний не лише прибирати, але й взаємодіяти з іншими елементами домашньої інфраструктури. Проте більшість комерційних моделей мають закрите програмне забезпечення, що обмежує можливості персоналізації, автоматизації та інтеграції зі сторонніми платформами. Саме тому виникає потреба у дослідженні відкритих програмних рішень для модифікації та розширення функцій цих пристроїв.

Особливий інтерес становить створення інструментів для кастомізації голосових повідомлень роботів-пилососів. Мова — це не лише зручний спосіб взаємодії, а й важливий елемент персоналізації, що підсилює відчуття комфорту та контролю. Платформи на зразок Home Assistant відкривають широкі можливості для такої інтеграції, дозволяючи перетворити простий пристрій у гнучкий компонент розумного середовища.

Мета дослідження

Метою дипломної роботи є розробка методів модифікації роботів-пилососів для глибшої інтеграції у систему розумного будинку, зокрема створення платформи для кастомізації голосових пакетів на базі відкритих програмних рішень.

Завдання дослідження: Щоб досягти поставленої мети, необхідно дослідити архітектуру типових моделей роботів-пилососів, вивчити можливості їх перепрошивки або розширення функцій через зовнішні платформи. Важливим кроком є аналіз існуючих голосових систем та форматів файлів, які використовуються в комерційних прошивках, і розробка інструментів для їх модифікації. Також слід спроєктувати веб-інтерфейс або платформу для зручного створення, редагування та завантаження голосових пакетів. Завершальний етап

включає інтеграцію модифікованого пристрою з системою на зразок Home Assistant, тестування роботи голосових повідомлень та оцінку користувацького досвіду.

Об’єкт дослідження: Об’єктом дослідження є роботизовані пилососи як частина системи розумного будинку.

Предмет дослідження: Предметом дослідження є програмне забезпечення роботів-пилососів, можливості його модифікації, а також процес створення та впровадження кастомізованих голосових повідомлень.

Наукова новизна: Наукова новизна полягає у розробці підходу до персоналізації побутових роботів за допомогою кастомізованих голосових пакетів, що можуть бути інтегровані у розумні системи. Запропонована платформа дозволяє користувачам без глибоких технічних знань змінювати голосові повідомлення пристроїв, створюючи новий рівень персоналізації та взаємодії в межах розумного будинку. Дослідження також демонструє нові можливості відкритих рішень у контексті побутової робототехніки.

1 ЗРОСТАННЯ ПОПУЛЯРНОСТІ РОБОТІВ-ПИЛОСОСІВ У ПОБУТІ

Роботи-пилососи стрімко змінюють спосіб, яким люди доглядають за своїми домівками. Ще кілька років тому вони здавалися чимось футуристичним, доступним лише для обраних. Сьогодні ж ці пристрої стали невід'ємною частиною багатьох домогосподарств, допомагаючи людям економити час і сили.

Одна з головних причин їхньої популярності — це зручність. Уявіть собі: вам більше не потрібно вручну тягати важкий пилосос по кімнатах, шукати розетки чи витрачати час на прибирання. Робот-пилосос самостійно виконує всю роботу — достатньо лише натиснути кнопку або дати голосову команду. Завдяки інтеграції зі смарт-системами, такими як Amazon Alexa чи Google Assistant, ви можете навіть не бути вдома, а ваш пристрій все одно забезпечить чистоту.

Технологічний прогрес зробив роботи-пилососи розумними. Сучасні моделі оснащені штучним інтелектом, який дозволяє їм запам'ятовувати планування приміщення, уникати перешкод і навіть адаптуватися до різних типів підлогових покриттів. Вони можуть автоматично регулювати потужність всмоктування залежно від того, чи прибирають килим чи тверду підлогу.

Ще один важливий аспект — це економія часу. У сучасному світі багато людей працюють довгі години або мають насичений графік. Робот-пилосос стає справжнім рятівником для тих, хто хоче підтримувати чистоту без зайвих зусиль. Він може працювати вночі або під час вашої відсутності, а деякі моделі навіть повертаються на зарядну станцію і продовжують прибирання після зарядки.

Популярність роботів-пилососів також зростає через їхню доступність. Раніше вони коштували дорого, але сьогодні на ринку є моделі на будь-який бюджет — від базових до преміальних із функцією миття підлоги та автоматичного очищення контейнера для сміття. Інтернет-магазини зробили їх ще доступнішими: користувачі можуть порівнювати ціни, читати відгуки і замовляти пристрої з доставкою додому.

Не можна забувати і про вплив екологічних факторів. Багато сучасних роботів-пилососів створено з використанням енергоефективних технологій та матеріалів, що сприяє зменшенню негативного впливу на довкілля.

Завдяки всім цим перевагам роботи-пилососи стали не просто модною новинкою, а справжнім помічником у домашніх справах для мільйонів людей по всьому світу. Із розвитком технологій їхній функціонал тільки розширюється, що робить ці пристрої ще більш корисними та потрібними у сучасному житті.

1.1 Відкриті програмні рішення для розширення функціональності роботів-пилососів

Відкриті програмні платформи та інструменти стали невід'ємною частиною розвитку побутової робототехніки. Вони надають користувачам, розробникам і дослідникам можливість не лише розширювати функціональність пристроїв, але й отримувати повний контроль над логікою їх роботи, зберігаючи при цьому незалежність від хмарних сервісів виробника. Такий підхід дозволяє гнучко налаштовувати поведінку робота-пилососа, інтегрувати його у локальну мережу розумного дому, реалізовувати кастомні сценарії, що враховують особисті потреби користувача або особливості планування приміщень. Усе це створює умови для формування відкритої екосистеми, де техніка легко адаптується до нових викликів, а користувач отримує більше свободи дій. [1]

ROS (Robot Operating System): це одна з найважливіших і найвпливовіших платформ у галузі відкритої робототехніки. ROS не є традиційною операційною системою, як, наприклад, Linux, але виступає в ролі фреймворку для побудови роботизованих систем із модульною архітектурою. Завдяки великій кількості готових пакетів, ROS дозволяє реалізувати управління рухом, взаємодію з сенсорами, картографування, навігацію та інші функції. Платформа активно використовується у дослідницькому середовищі та в реальних прикладних розробках, зокрема у створенні власних версій роботів-пилососів. Велика перевага ROS полягає у її підтримці емуляційного середовища Gazebo, що дозволяє безпечно тестувати алгоритми до їх впровадження на фізичному пристрої.

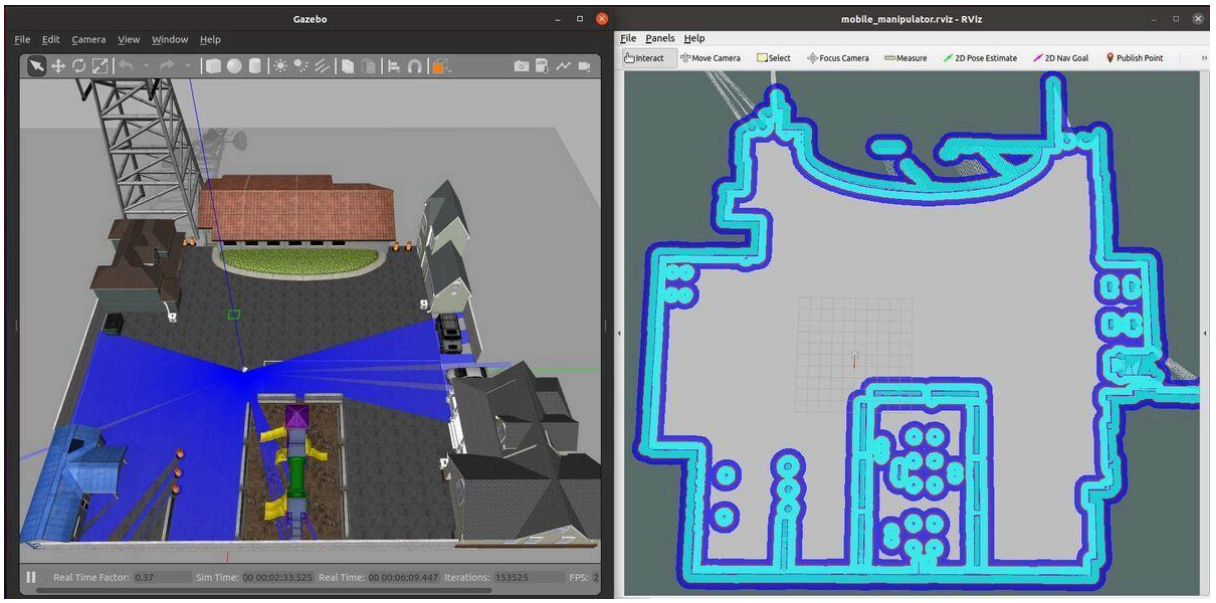


Рисунок 1.1 Симуляція руху робота-пилососа з операційною системою ROS

OpenCV: це бібліотека комп’ютерного зору з відкритим кодом, яка широко використовується у сфері робототехніки. Її можливості дозволяють реалізовувати розпізнавання об’єктів, виявлення перешкод, аналіз просторових карт, і навіть розрізнення типів поверхонь. У контексті роботів-пилососів OpenCV може застосовуватися для побудови віртуальної карти простору, виявлення дверей, меблів або навіть зон із високим рівнем забруднення. Завдяки цьому робот може адаптувати свою поведінку до умов конкретного приміщення, забезпечуючи ефективніше та точніше прибирання. [2]

Valetudo: це проєкт, що виник як відповідь на потребу користувачів у локальному управлінні роботами-пилососами без використання хмарних серверів виробників. Цей інструмент дозволяє повністю контролювати пристрій за допомогою веб-інтерфейсу або спеціального додатку, а також підтримує інтеграцію з популярними системами автоматизації, такими як Home Assistant, через MQTT-протокол. Valetudo працює на прошивках роботів, що використовують прошивку з Linux-подібною системою, і надає інтерфейс для відображення карти прибирання, вибору зон і встановлення розкладів. Його відкритість і прозорість роблять цей проєкт привабливим як для досвідчених

розробників, так і для звичайних користувачів, які цінують приватність та автономність. [3]

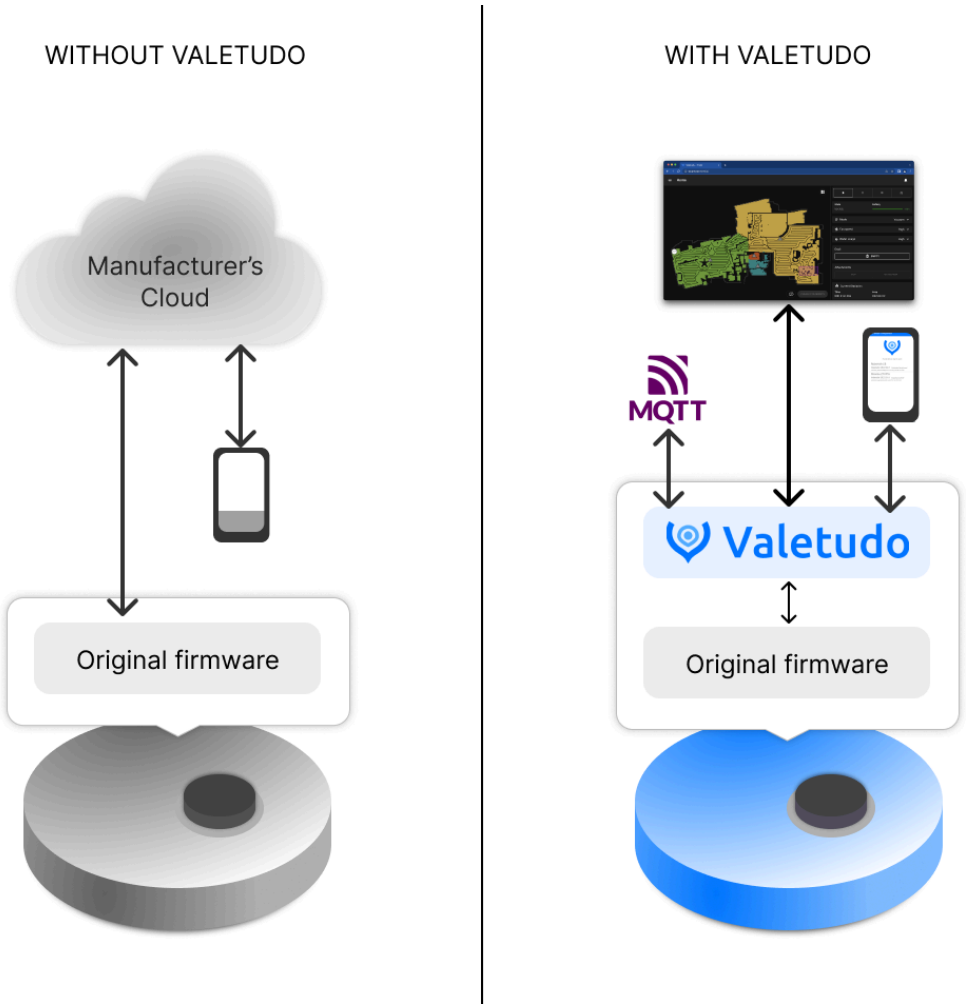


Рисунок 1.2 Хмарне та локальне керування роботом пилососом

Congatudo: це ще один відкритий проєкт, подібний до Valetudo, але орієнтований на підтримку роботів бренду Conga. Його створено для локального керування без участі сторонніх серверів, і він також підтримує такі функції, як сегментація приміщень, створення карт та автоматизація процесів за допомогою MQTT. Congatudo дозволяє адаптувати функціональність під особливості конкретних моделей роботів, що значно спрощує інтеграцію з іншими системами та розширює можливості кастомізації. [4]

Webots: це платформа для симуляції роботизованих систем, яка дає змогу створювати віртуальні моделі пилососів, тестувати алгоритми руху та поведінки у штучному середовищі. Webots підтримує програмування на мовах Python, C++,

Java та MATLAB, що дозволяє легко включити його у процес розробки як студентам, так і професійним інженерам. За допомогою цього середовища можна побудувати цифрового двійника робота, протестувати сценарії картографування, об’їзду перешкод, або поведінки на різних типах підлоги без ризику для фізичного обладнання.

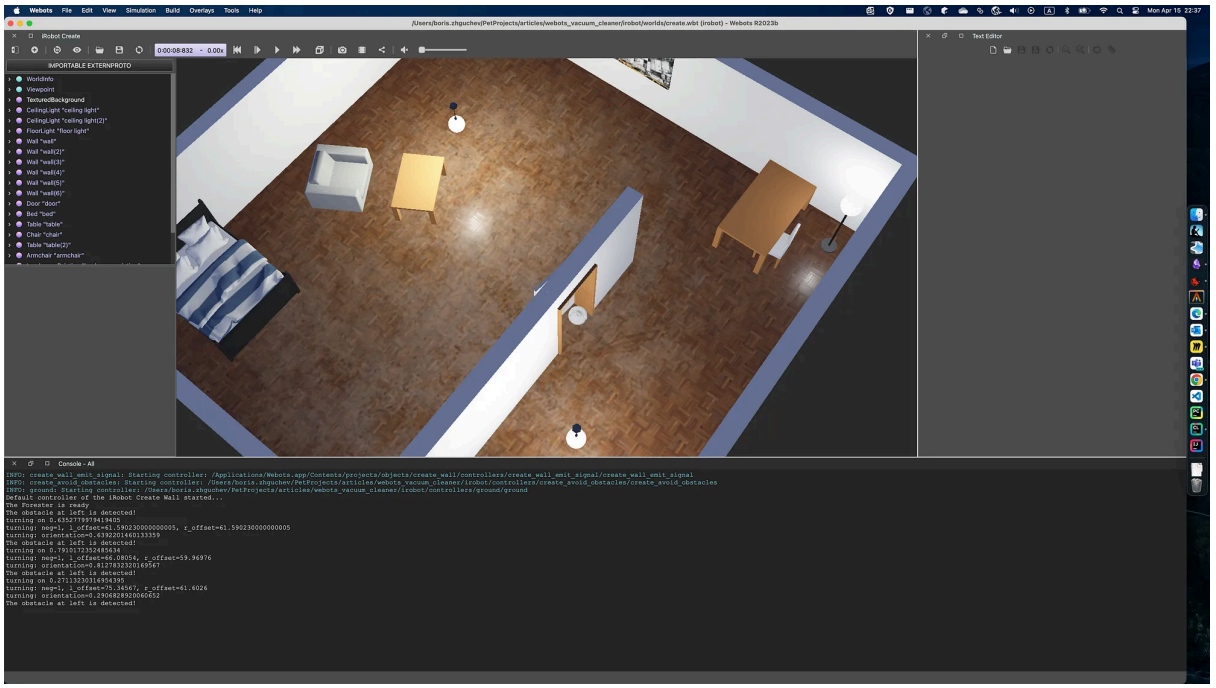


Рисунок 1.3 Симуляція руху робота-пилососа в ПЗ Webots та Forester

Arduino та Raspberry Pi: ці платформи залишаються фундаментальними для прототипування та побудови повноцінних кастомних рішень. Arduino забезпечує зручну роботу з сенсорами, актуаторами та простими алгоритмами, тоді як Raspberry Pi дозволяє розгорнути повноцінну операційну систему, запускати веб-сервери, обробляти зображення, працювати з ROS або OpenCV. Разом ці платформи утворюють потужний інструментарій для створення гнучких, розширюваних систем, що можуть повністю замінити або суттєво вдосконалити функціональність комерційного робота-пилососа. [5]

Таким чином, відкриті інструменти дозволяють не тільки повторити функціональність готових пристроїв, а й вийти за її межі, надаючи можливість створювати більш ефективні, адаптивні та безпечні рішення. Вони сприяють демократизації технологій, навчанню, експериментуванню та, зрештою,

відкривають шлях до більш прозорої, контрольованої та персоналізованої побутової автоматизації.

1.2 Переваги й недоліки відкритих платформ для роботів-пилососів

Розвиток відкритих рішень у сфері побутової робототехніки, зокрема роботів-пилососів, відкриває перед користувачами та розробниками нові можливості, які раніше були доступні лише виробникам. Такі платформи, як Valetudo, ROS чи Congatudo, не просто надають контроль над пристроєм, а змінюють саму парадигму взаємодії з технікою. Однією з найвагоміших переваг є незалежність від хмарних сервісів. Коли пристрій працює повністю локально, користувач отримує контроль над зібраними даними, їхнім зберіганням і використанням. Це особливо важливо для тих, хто цінує конфіденційність, адже більшість заводських систем передають інформацію на сервери виробників. Відкриті рішення дозволяють уникнути цієї залежності й будувати локальну інфраструктуру, яка працює автономно. [6]

Ще однією перевагою є розширення функціональності. За допомогою відкритих платформ можна не лише змінювати поведінку робота, а й додавати функції, яких не передбачено в офіційному програмному забезпеченні. Наприклад, користувач може реалізувати автоматичне прибирання лише в певних зонах, створювати інтеграції з іншими пристроями у системі розумного будинку, або навіть додавати підтримку голосових пакетів іншими мовами. Відкритий код дозволяє краще зрозуміти, як саме працює пристрій, і адаптувати його під власні потреби.

Крім того, відкриті рішення стимулюють навчання та експерименти. Вони заохочують розробників, студентів та ентузіастів досліджувати, як влаштовані системи навігації, картографії, керування. Робота з такими платформами дає змогу опанувати сучасні технології, навчитися працювати з сенсорами, обробкою сигналів, мережею та автоматизацією. У навчальних закладах або хакерспейсах відкриті прошивки та інструменти стають основою для проведення експериментів і створення прототипів. [7]

Проте, попри численні переваги, відкриті платформи мають і свої недоліки. Одним із найсуттєвіших обмежень є вузький перелік моделей, які можна перепрошити або інтегрувати в подібні рішення. Найчастіше це пристрої певних брендів і версій, які мають специфічні апаратні або програмні особливості. Це значно звужує вибір для пересічного користувача і вимагає попереднього вивчення сумісності. Також існує ризик втрати офіційної гарантії. Більшість виробників не підтримують зміну прошивки і вважають це втручанням у роботу пристрою. У разі несправності користувач може залишитися без сервісної підтримки. Це змушує зважувати ризики перед тим, як розпочати модифікацію. [8]

Ще один фактор — це поріг входу. Хоча відкриті проєкти зазвичай мають активні спільноти та документацію, все ж потрібно мати базове технічне розуміння. Знання про мережі, Linux, MQTT чи навіть просто навички роботи з командним рядком стають необхідними для налаштування та використання таких рішень. Для недосвідчених користувачів це може стати бар’єром або навіть причиною для відмови від використання відкритої платформи. [9]

Попри ці обмеження, розвиток відкритих рішень для роботів-пилососів є перспективним напрямом, що поступово змінює уявлення про функціональність побутових роботів. Вони дають змогу користувачам не лише адаптувати пристрій під власні потреби, а й долучатися до спільноти, яка постійно вдосконалює доступні інструменти. Таким чином, відкриті рішення формують альтернативу закритим екосистемам, роблячи технології доступнішими, зрозумілішими та більш гнучкими. [10]

1.3 Потреба у забезпеченні конфіденційності даних, які збирають роботи-пилососи

Роботи-пилососи за останнє десятиліття стали невід’ємною частиною сучасного побуту, проте з їх широким розповсюдженням зросли й ризики, пов’язані з обробкою персональних даних. Попри те, що основна функція цих пристроїв полягає в автоматичному прибиранні житлового простору, сучасні

моделі оснащені численними сенсорами, камерами та модулями навігації, які активно збирають і обробляють візуальну, просторову й поведінкову інформацію.

У низці випадків виробники не надають повного контролю над зібраними даними, зберігаючи або обробляючи їх у хмарних сервісах. Це створює потенційні ризики, пов’язані як з недобросовісним використанням інформації, так і з можливими витоками внаслідок технічних вразливостей. Існують прецеденти, коли інформація, зібрана роботами-пилососами, ставала доступною для сторонніх осіб. Один із найбільш відомих випадків — інцидент із пристроєм Deebot, виробленим компанією Ecovacs. У 2022 році через вразливість у системі безпеки зловмисники отримали доступ до фотографій, зроблених вбудованими камерами роботів, зокрема зображень з інтимних зон приватного житла. Ці матеріали згодом були поширені в мережі, що викликало широкий суспільний резонанс і поставило під сумнів безпеку подібних технологій.

У травні 2024 року було зафіксовано ще один серйозний випадок, коли хакери використали Bluetooth-з’єднання для віддаленого доступу до пристроїв тієї ж компанії. Зловмисники отримали можливість переглядати відео з камер у реальному часі та навіть транслювати образливі повідомлення через вбудовані динаміки. Цей інцидент вкотре підтвердив, що недоліки в захисті одного компонента можуть стати каналом доступу до всієї домашньої мережі. [11]

Окрім хакерських атак, ризик становить і недобросовісне використання даних з боку самих виробників. Наприклад, дані про планування приміщень, тип меблів або частоту використання окремих кімнат можуть бути використані для таргетованої реклами або передані третім сторонам без належної поінформованості кінцевого користувача.

Для зниження ризиків деякі розробники пропонують альтернативне програмне забезпечення, що дозволяє повністю відмовитися від хмарних сервісів. Одним із таких рішень є відкритий проєкт Valetudo, який забезпечує локальне зберігання даних і повний контроль над пристроєм без передачі інформації до зовнішніх серверів. [12]

У контексті розвитку індустрії розумного дому забезпечення конфіденційності має стати пріоритетом як для виробників, так і для споживачів. Відсутність прозорості у використанні персональних даних, як і недосконалий захист пристроїв, може спричинити серйозні порушення приватності та призвести до репутаційних, а в деяких випадках — і правових наслідків.

1.4 Значення відкритих програмних рішень для модифікації функцій роботів

Відкриті програмні рішення стали справжньою революцією у світі робототехніки, зокрема для роботів-пилососів. Вони відкрили нові можливості для користувачів, які прагнуть адаптувати пристрої під свої унікальні потреби. Завдяки таким рішенням роботи-пилососи перестали бути просто "чорними ящиками", функціонал яких обмежується задумами виробника. Тепер кожен може налаштувати свій пристрій так, щоб він працював максимально ефективно і зручно. [13]

Однією з головних переваг відкритого програмного забезпечення є свобода кастомізації. Наприклад, за допомогою альтернативних прошивок, таких як Valetudo, користувачі можуть відмовитися від використання хмарних сервісів і зберігати всі дані локально. Це не лише покращує конфіденційність, але й дозволяє управляти пристроєм без підключення до інтернету. Для багатьох людей це важливий крок до створення більш безпечного "розумного дому".

Ще одним вагомим аспектом є можливість додавання нових функцій. Наприклад, ентузіасти можуть змінювати алгоритми навігації робота, налаштовувати голосові повідомлення або інтегрувати пристрій із системами автоматизації, такими як Home Assistant. Це дозволяє створювати сценарії, які роблять робота частиною складнішої екосистеми: він може починати прибирання після того, як ви залишили дім, або зупинятися, якщо ви повернулися. [14]

Відкриті програмні рішення також сприяють розвитку спільнот розробників і користувачів. Люди діляться своїми напрацюваннями, створюють нові модулі та вдосконалюють існуючі функції. Це дає змогу навіть недосвідченим користувачам скористатися готовими рішеннями для покращення роботи свого пристрою.

Крім того, відкритість програмного забезпечення стимулює конкуренцію серед виробників. Користувачі все частіше обирають пристрої з можливістю кастомізації, що змушує компанії враховувати ці потреби і пропонувати більш гнучкі продукти. [15]

Загалом, відкриті програмні рішення — це не лише про технічний прогрес, але й про свободу вибору. Вони дозволяють користувачам брати активну участь у розвитку технологій і створювати пристрої, які відповідають їхнім індивідуальним потребам. У світі роботів-пилососів це означає більше можливостей для інновацій і персоналізації, що робить ці пристрої ще кориснішими та цікавішими для широкого кола людей.

2 VALETUDO: АВТОНОМІЗАЦІЯ РОБОТІВ-ПИЛОСОСІВ НА ОСНОВІ ВІДКРИТОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Valetudo — це програмне забезпечення з відкритим кодом, розроблене для роботи з певними моделями роботів-пилососів, яке дозволяє повністю відмовитися від використання хмарних сервісів виробника. Замість того щоб надсилати дані про прибирання на віддалені сервери, робот із Valetudo зберігає всю інформацію локально, тобто всередині домашньої мережі користувача. Такий підхід особливо цінний з огляду на питання конфіденційності, безпеки та стабільності роботи пристрою. [16]

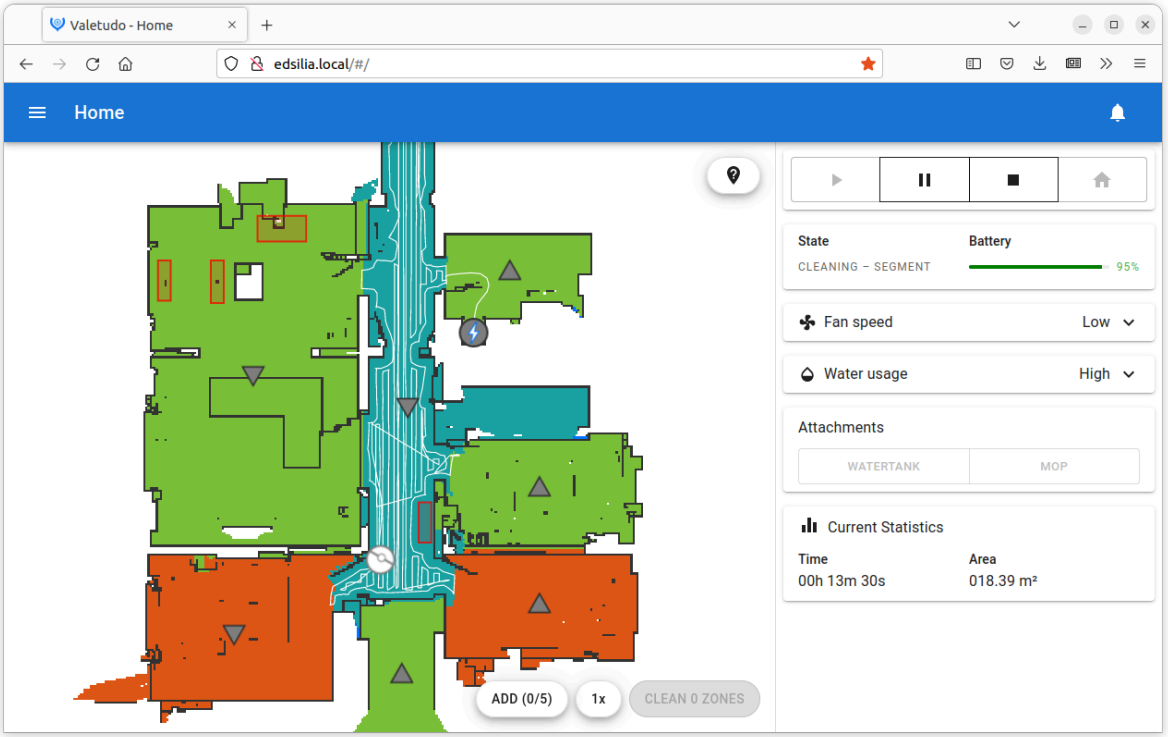


Рисунок 2.1 Графічний інтерфейс для керування прошивкою Valetudo

Більшість сучасних роботів-пилососів працюють у тісній прив’язці до інтернету, надсилаючи інформацію про карту приміщення, маршрути прибирання, графіки запуску й навіть голосові команди на сервери виробника. Це може створювати додаткові ризики — від витоку особистих даних до неможливості користування пристроєм у разі втрати інтернет-з’єднання або закриття сервісу

самим виробником. Valetudo пропонує альтернативу — автономну систему, яка виконує всі основні функції без зовнішніх підключень. [17]

Програмне забезпечення Valetudo встановлюється безпосередньо на сам робот, зазвичай після модифікації його прошивки. Це вимагає певного технічного рівня підготовки, оскільки процес часто включає доступ до внутрішньої операційної системи пристрою, роботу з командною оболонкою Linux, використання послідовного порту UART, а в деяких випадках навіть розбирання корпусу. Проте в мережі існує чимало докладних інструкцій та підтримки з боку спільноти, що суттєво полегшує цей процес. [18]

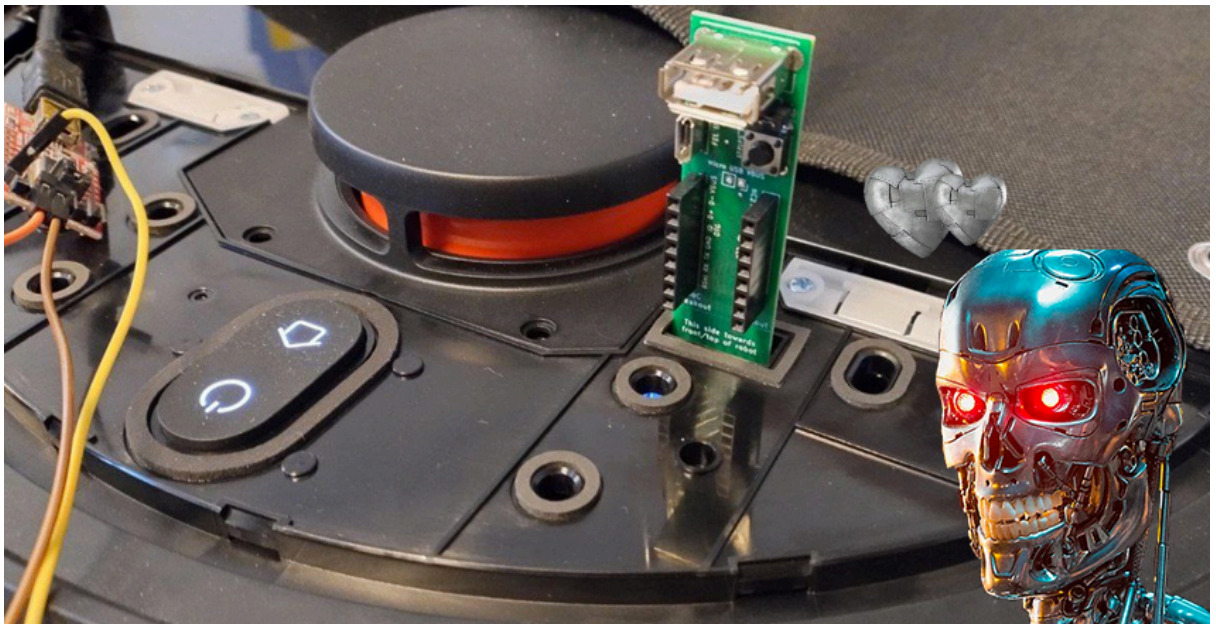


Рисунок 2.2 UART перехідник для прошивки робота-пилососа ПЗ Valetudo

Особливість Valetudo полягає в тому, що воно не лише відключає доступ до інтернету, а й надає користувачеві зручний інтерфейс для керування роботом. Через браузер можна переглядати карту прибирання, позначати віртуальні стіни, зони, які потрібно або не потрібно прибирати, запускати та зупиняти прибирання, а також переглядати статистику. Цей інтерфейс працює напряму з самим пристроєм, без участі зовнішніх серверів. [19]

Valetudo підтримує інтеграцію з такими платформами, як Home Assistant, MQTT, ioBroker, що дозволяє будувати складні сценарії автоматизації. Наприклад, робот може автоматично починати прибирання, коли всі мешканці покидають дім, або припиняти його, якщо активується сигналізація. Завдяки відкритому коду,

будь-хто може дописати власні модулі або змінити логіку роботи пристрою під свої потреби. [20]

Назва “Valetudo” походить з латинської мови та означає “здоров’я” або “сила”. Це символічно, адже програмне забезпечення надає пристрою “нове життя”, звільняючи його від обмежень, накладених виробником. Проєкт має активну спільноту розробників і користувачів, які постійно вдосконалюють функціональність, адаптують Valetudo до нових моделей роботів і створюють детальну документацію.

Таким чином, Valetudo є важливим інструментом у контексті модифікації та кастомізації роботів-пилососів. Воно не лише розширює технічні можливості пристрою, а й повертає контроль над його роботою безпосередньо користувачеві, що є ключовим у філософії відкритого програмного забезпечення. [21]

З практичної точки зору використання Valetudo також дозволяє значно зменшити навантаження на мережу та споживання трафіку, що може бути важливо у випадках із нестабільним або обмеженим інтернетом. Відсутність зв’язку з хмарою також означає, що робот не отримає автоматичних оновлень, які іноді можуть змінити або обмежити його функціональність. Користувач сам вирішує, коли і які зміни вносити до системи, тим самим уникаючи нав’язаних змін з боку виробника. [22]

Ще однією цікавою перевагою є можливість глибшого аналізу роботи пристрою. Завдяки відкритому доступу до логів, телеметрії та карт прибирання користувач може вивчити, як саме поводиться робот у різних умовах, і відповідно вдосконалити маршрути або логіку прибирання. Усе це робить Valetudo не просто альтернативним інтерфейсом, а інструментом для побудови повноцінного інтелектуального рішення в межах персонального розумного дому.

2.1 Congatudo: локальний інтерфейс для керування роботами Conga

Congatudo — це проєкт з відкритим вихідним кодом, який надає користувачеві можливість повністю локального управління роботами-пилососами серії Conga без потреби у використанні хмарних сервісів виробника. Це рішення

не є альтернативною прошивкою або низькорівневим програмним забезпеченням. Congatudo не змінює поведінку або логіку роботи пристрою, а натомість виступає як інтерфейс взаємодії з уже наявною прошивкою. Він запускається на окремому пристрої, як-от локальному сервері або навіть звичайному комп'ютері, і через протоколи локального зв'язку дозволяє контролювати робота з браузера або мобільного пристрою. [23]

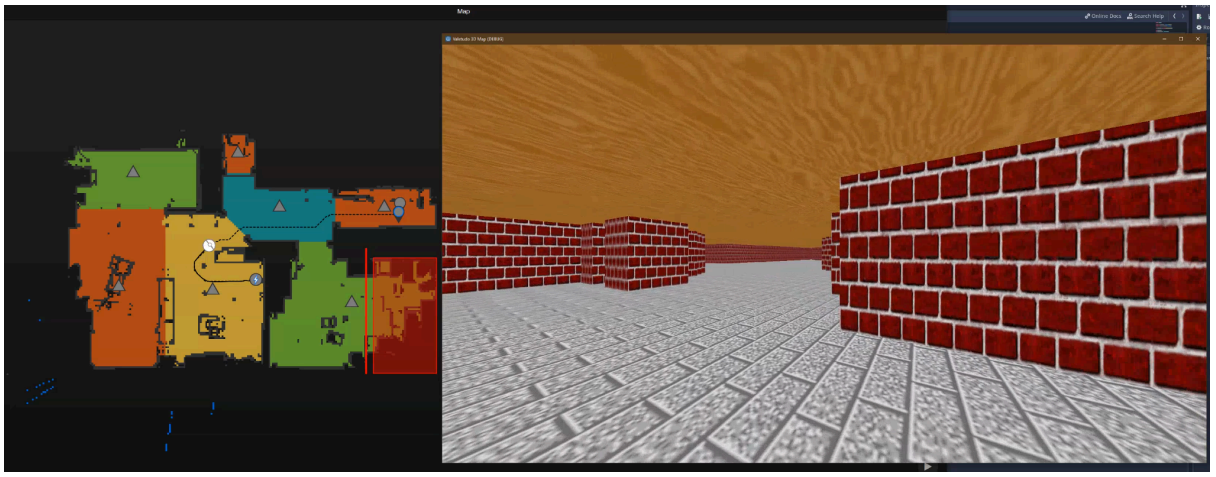


Рисунок 2.3 Графічний інтерфейс керування ПЗ Congatudo

З технічного боку, Congatudo базується на проєкті Valetudo, що давно зарекомендував себе як ефективна альтернатива хмарному управлінню для роботів Ecovacs, Roborock та інших. Подібно до нього, Congatudo побудований з урахуванням високих вимог до приватності, автономності та інтеграції з іншими системами автоматизації. Проєкт підтримує MQTT — легкий протокол обміну повідомленнями, який широко використовується в розумних будинках. Завдяки цьому, робота-пилососа можна інтегрувати до таких систем, як Home Assistant або Node-RED, що відкриває широкі можливості для створення складних сценаріїв автоматизації, зокрема запуску при виході з дому або в непікові години електроспоживання. [24]

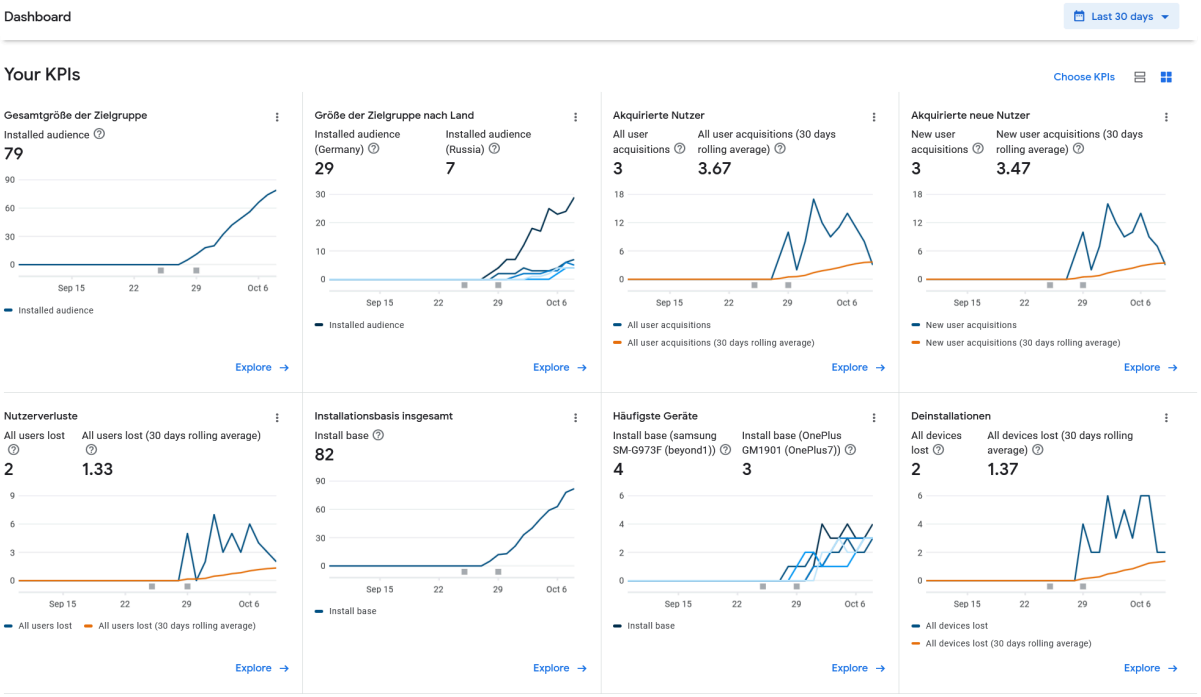


Рисунок 2.4 статистика використання робота пилососа в ПЗ Congatudo

Congatudo має адаптивний вебінтерфейс, доступний з будь-якого сучасного браузера, незалежно від платформи. Він дозволяє запускати або зупиняти прибирання, переглядати карту приміщення, контролювати рівень заряду батареї, слідкувати за станом витратних матеріалів та створювати зони прибирання. При цьому управління повністю відбувається в межах локальної мережі, без передачі даних на зовнішні сервери, що особливо важливо для користувачів, які цінують приватність або мають обмежене або нестабільне підключення до Інтернету. [25]

На відміну від Valetudo, який підтримує ширший спектр моделей і в деяких випадках вимагає модифікації прошивки, Congatudo зосереджений виключно на моделях бренду Conga і працює з уже наявною прошивкою пристрою. Це значно спрощує процес встановлення для новачків і зменшує ризики, пов’язані з втручанням у низькорівневе ПЗ. Для запуску достатньо мати сумісного робота та пристрій для розміщення Congatudo, наприклад Raspberry Pi або міні-ПК, на якому запуститься контейнер Docker або звичайний Node.js сервер.

Розробник проєкту, відомий під псевдонімом elraro, активно підтримує розробку, публікує оновлення та веде документацію, в тому числі покрокові інструкції з налаштування. Серед додаткових інструментів також доступні

мобільні застосунки-компаньйони, які полегшують доступ до функціоналу Congatudo з Android-пристроїв. [26]

Таким чином, Congatudo — це сучасне, легке у використанні рішення, що значно розширює можливості керування роботами-пилососами Conga для ентузіастів та користувачів, які прагнуть автономності, локального контролю та конфіденційності. Його використання дає змогу уникнути прив’язки до зовнішніх серверів, водночас забезпечуючи зручний і функціональний інтерфейс для щоденної експлуатації робота.

2.2 Протокол Xiaomi Miot

Для забезпечення обміну даними між пристроями екосистеми Xiaomi розроблений спеціальний протокол зв’язку. На початкових етапах використовувався протокол miio, який характеризувався досить простою структурою запитів. Проте документація до нього офіційно не публікувалася, що ускладнювало розробку сторонніх рішень, оскільки команди доводилося відновлювати шляхом зворотного аналізу трафіку. [27]

З метою впорядкування взаємодії та підвищення доступності інформації була створена нова система під назвою Miot. Вона базується на відкритій специфікації, де для кожного пристрою детально описані його функціональні можливості, параметри та допустимі дії. Наприклад, для роботизованого пилососа у специфікації буде вказано можливість запуску прибирання, повернення на зарядну станцію, зміну мови голосового супроводу тощо.

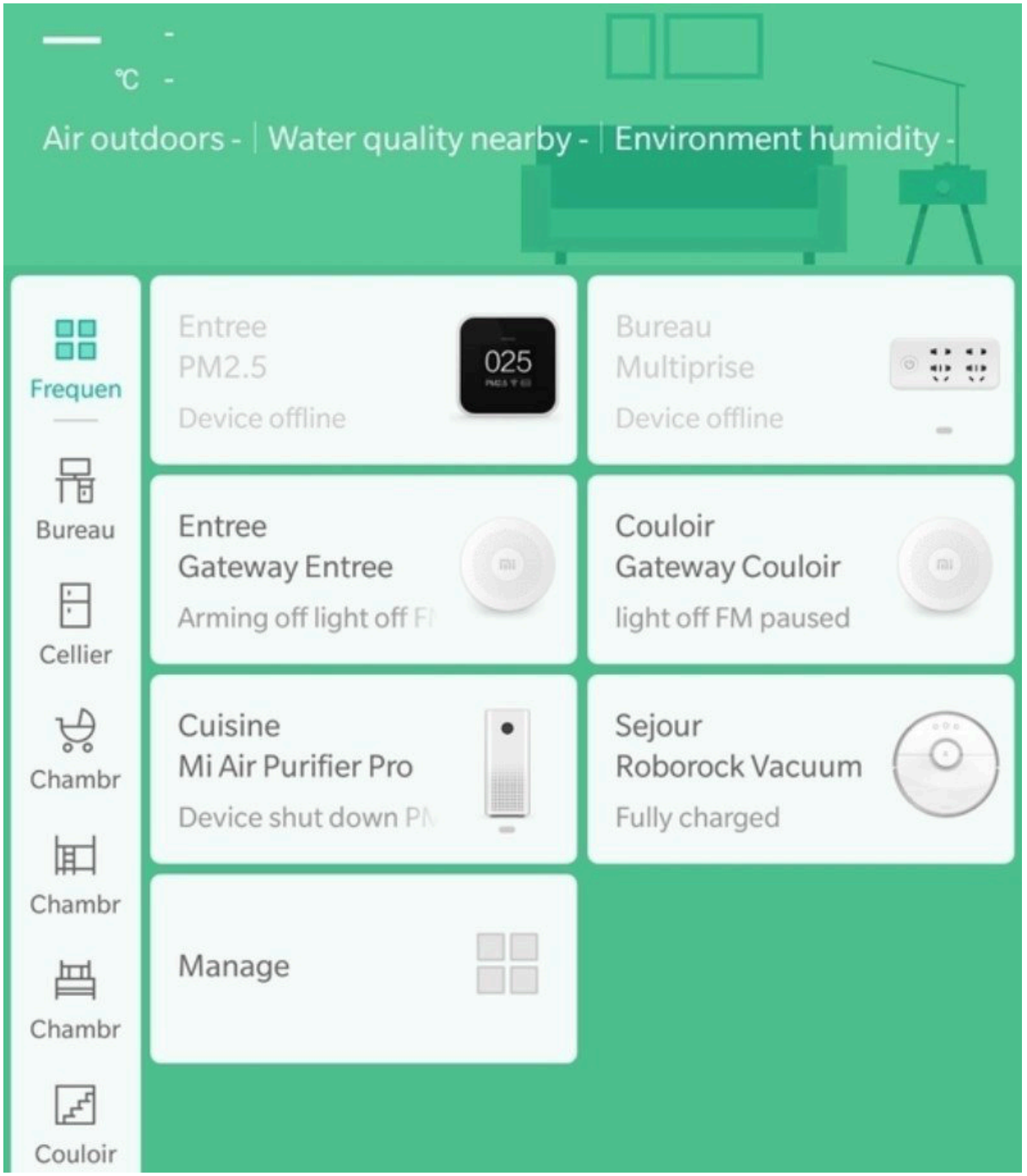


Рисунок 2.5 Пристрої, що керуються протоколом Xiaomi Miot

Повна документація доступна у вільному доступі на сайті home.miot-spec.com. У ній кожен пристрій представлений як набір сервісів і властивостей, де кожен сервіс (наприклад, “управління аудіо”) і кожна властивість (наприклад, “поточний рівень заряду акумулятора”) має свій унікальний ідентифікатор: siid для сервісу та piid для властивості. [28]

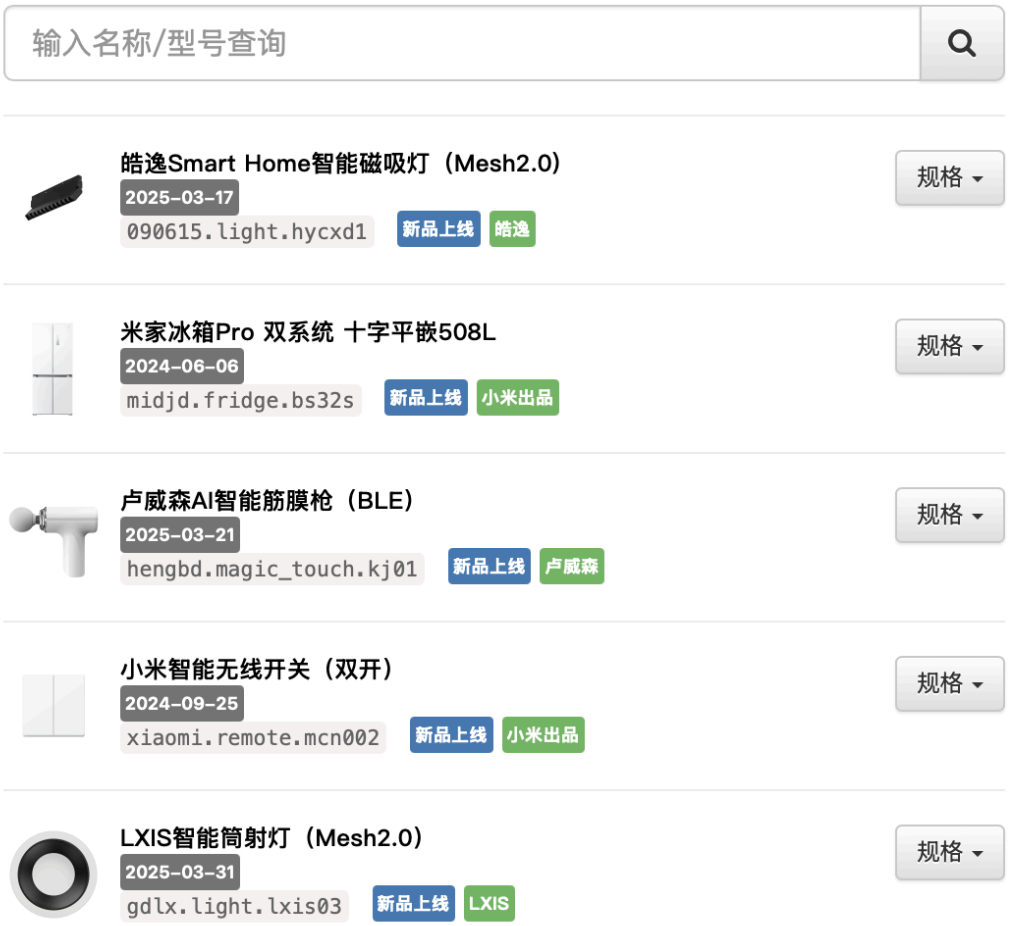


Рисунок 2.6 Специфікація Xiaomi пристроїв на сайті home.miot-spec.com

При взаємодії із пристроєм необхідно передавати ці ідентифікатори разом із даними запиту. Наприклад, під час надсилання команди для зміни голосового пакета слід вказати правильні siid та piid разом із посиланням на відповідний аудіофайл. У разі невірного формування запиту пристрій його не розпізнає або проігнорує.

Хоча специфікація Miot є відкритою, сам протокол має досить складну структуру, що вимагає точності у формуванні запитів. Для полегшення роботи з пристроями розроблено проєкт python-miio — відкритий (опенсорсний) набір бібліотек на мові програмування Python. Цей проєкт реалізує підтримку як старого протоколу miio, так і нового Miot, дозволяючи надсилати команди до пристроїв Xiaomi без необхідності створення власної реалізації протоколу з нуля.

Порівняно з попереднім протоколом miio, де структура команд була досить хаотичною і незадокументованою, протокол Miot пропонує систематизований і

стандартизований підхід до керування пристроями. Завдяки цьому стало можливим ефективніше інтегрувати пристрої Xiaomi у сторонні системи розумного дому, такі як, наприклад, Home Assistant.

2.3 DustCloud та DustBuilder: локальне керування роботами-пилососами

Багато сучасних роботів-пилососів, таких як пристрої Xiaomi, Roborock або Dreame, розраховані на роботу через хмарні сервіси виробника. Це означає, що для їх повноцінного функціонування необхідне підключення до інтернету та постійне обслуговування через зовнішні сервери компанії. Однак для користувачів, які прагнуть більшого контролю над своїми пристроями, важливо мати можливість локального керування без залучення сторонніх серверів. Саме для таких цілей були створені два важливі проєкти — DustCloud та DustBuilder. [29]

DustCloud — це проєкт з відкритим кодом, який займається зворотним інжинірингом хмарної платформи Xiaomi для розумних пристроїв. Мета DustCloud полягає у створенні альтернативи для хмарних сервісів, що дозволяє контролювати пристрої напряму через локальну мережу. У межах проєкту було ретельно проаналізовано, яким чином роботи-пилососи обмінюються даними з серверами Xiaomi, розібрано мережеві протоколи та принципи авторизації, а також емульовано необхідні сервіси для локального обслуговування пристроїв.

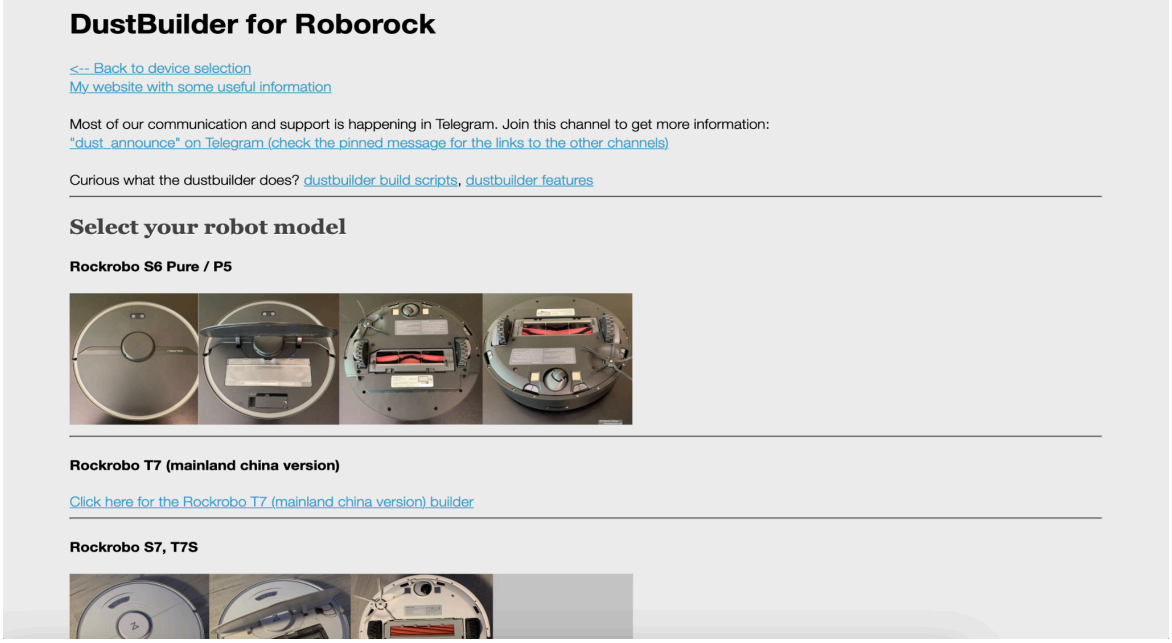


Рисунок 2.7 Вебсайт проєкту Dust Builder

Одним із важливих результатів DustCloud є можливість запуску локального сервера, який приймає запити пристрою так, ніби це офіційний сервер Xiaomi. У результаті пристрій вважає, що він працює через “хмару”, хоча насправді комунікація відбувається всередині домашньої мережі. Це дозволяє повністю контролювати робота-пилососа без зовнішніх підключень і без збору даних третіми особами. [30]

DustCloud надає набір інструментів, серед яких програми для перехоплення трафіку, скрипти для модифікації прошивок, а також утиліти для отримання root-доступу до пристроїв. Наприклад, однією з важливих утиліт є інструмент dustcloud для емуляції серверу авторизації.

```
dustcloud --config config.json
```

Ця команда дозволяє запустити локальний сервер, через який пристрій проходитиме початкову авторизацію.

Щоб повноцінно скористатися локальним керуванням, часто виникає потреба у зміні або перепрошивці пристрою. Саме для цього був створений інший проєкт — DustBuilder.

DustBuilder — це спеціальний онлайн-сервіс для автоматичної генерації кастомних прошивок для сумісних моделей роботів-пилососів. Використання

DustBuilder дозволяє створити прошивку, яка одразу включає необхідні зміни, наприклад доступ через SSH, root-доступ або встановлення альтернативного програмного забезпечення, такого як Valetudo. [31]

Valetudo — це ще один проєкт, спрямований на локальне керування роботами-пилососами. Він дозволяє налаштувати пристрій так, щоб він працював автономно через інтерфейс у браузері, не підключаючись до зовнішніх серверів.

На сайті DustBuilder користувач обирає модель свого пристрою зі списку підтримуваних, налаштовує потрібні параметри і запускає процес складання прошивки. Після завершення сервіс надає файл з розширенням .pkg, який потім можна встановити на пристрій стандартними методами. Приклад команди для встановлення оновлення за допомогою python-miio:

```
miio firmware_update --ip 192.168.1.100 --token your_token firmware.pkg
```

У цьому прикладі firmware.pkg — це файл кастомної прошивки, створеної за допомогою DustBuilder.

DustBuilder має низку переваг. По-перше, він автоматизує всі складні операції, які раніше вимагали ручного втручання і глибокого розуміння системи. По-друге, створена прошивка базується на офіційних образах, що гарантує сумісність з пристроями. По-третє, DustBuilder дозволяє безпечно інтегрувати нові можливості без ризику “зламати” пристрій неправильними діями. [32]

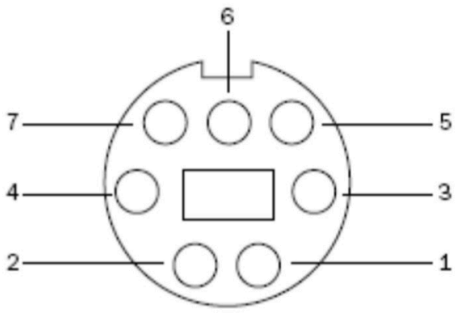
І DustCloud, і DustBuilder активно розвиваються силами відкритої спільноти ентузіастів. Це означає, що з часом підтримка нових моделей пристроїв розширюється, інструменти стають більш стабільними та зручними у використанні. І хоча використання цих проєктів передбачає певні технічні навички, завдяки докладній документації та численним прикладам процес стає доступним навіть для користувачів-початківців.

У підсумку DustCloud та DustBuilder відкривають можливості для повної автономії пристроїв розумного дому, дозволяють підвищити безпеку

персональних даних та розширити функціональні можливості комерційних продуктів без прив’язки до обмежень, накладених виробником. [33]

2.4 Open Interface: відкритий протокол для гнучкого керування роботами

Open Interface — це стандарт, розроблений компанією iRobot для забезпечення програмного доступу до роботів через послідовний порт. Цей протокол створений для того, щоб полегшити розробку і інтеграцію роботизованих рішень, надаючи зовнішнім пристроям можливість взаємодіяти з роботами та керувати їх функціоналом. Open Interface дозволяє програмістам і дослідникам змінювати та налаштовувати поведінку роботів, створюючи нові алгоритми для навігації, аналізу даних або інших операцій. [34]



Pin	Name	Description
1	Vpwr	Roomba battery + (unregulated)
2	Vpwr	Roomba battery + (unregulated)
3	RXD	0 – 5V Serial input to Roomba
4	TXD	0 – 5V Serial output from Roomba
5	BRC	Baud Rate Change
6	GND	Roomba battery ground
7	GND	Roomba battery ground

Рисунок 2.8 7-контактний роз'єм mini-DIN робота-пилососа Roomba

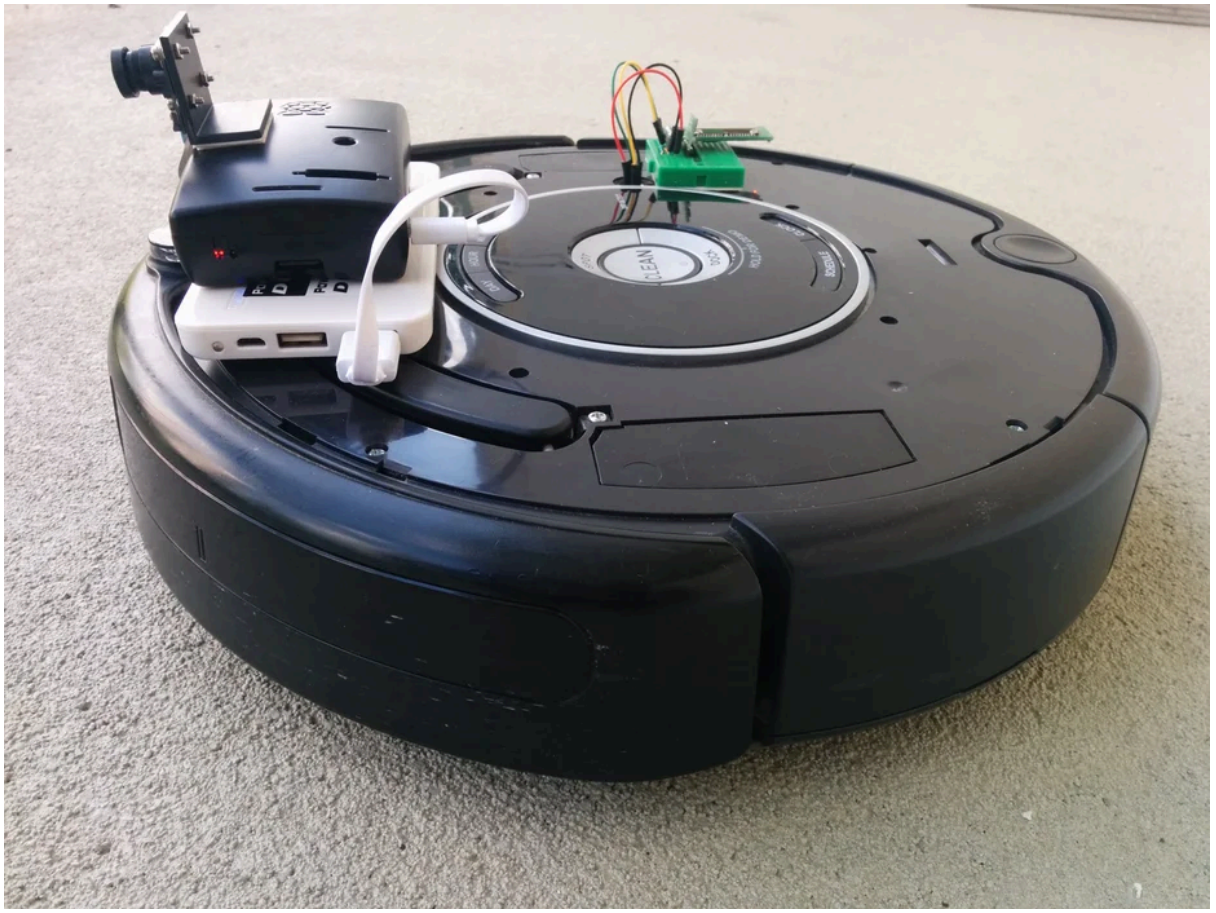


Рисунок 2.9 Керування пилососом Roomba за допомогою одноплатного комп’ютера Raspberry Pi [35]

В основі Open Interface лежить послідовний інтерфейс UART (Universal Asynchronous Receiver/Transmitter), через який можна надсилати команди та отримувати дані від робота. Це дозволяє підключати до робота різні контролери, наприклад, Arduino або Raspberry Pi, для взаємодії з його системами. Завдяки такому доступу можна управляти рухами робота, включати чи вимикати сенсори, збирати інформацію про стан батареї, а також отримувати дані з інших сенсорів, таких як датчики наявності перешкод, сенсори очищення та інші.

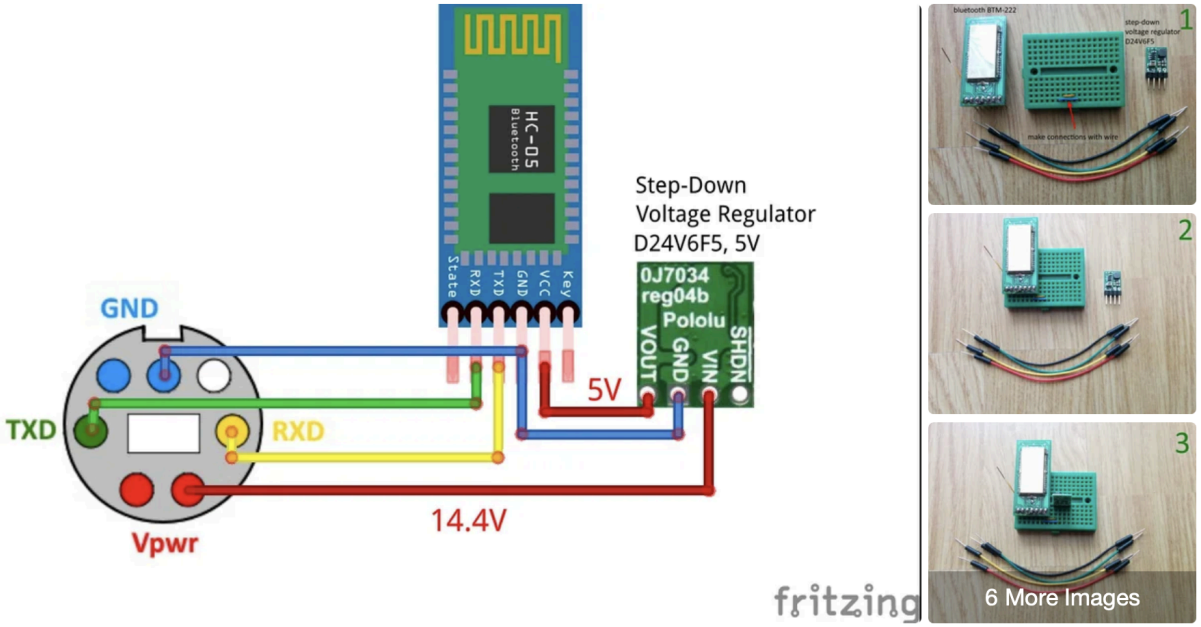


Рисунок 2.10 Приєднання Bluetooth сернсора HC-05 до mini-DIN роз'єму робота-пилососа Roomba

Open Interface є досить простим для розуміння, навіть для новачків. Протокол описує базові команди для керування роботом, включаючи рухи вперед, назад, повороти, а також спеціальні функції для взаємодії з датчиками. Цей стандарт не лише дозволяє керувати роботом, але й дає змогу взаємодіяти з ним на більш глибокому рівні, отримуючи зворотний зв'язок про поточний стан пристрою. Наприклад, можна відстежувати рівень заряду батареї, стан сенсорів, або навіть отримувати інформацію про те, в яких частинах приміщення робот працював або не працював. [36]

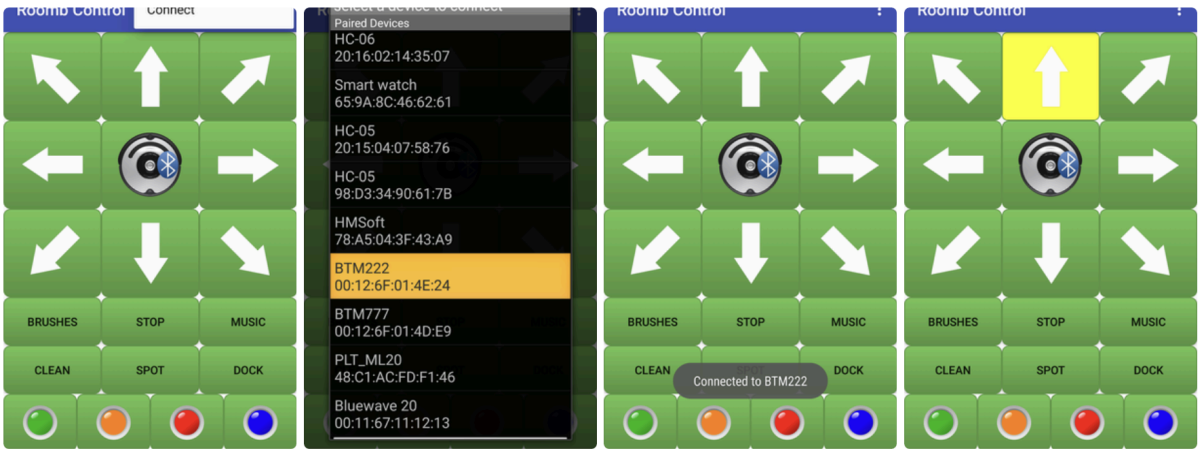


Рисунок 2.11 приклад керування роботом Roomba по Bluetooth

Open Interface також інтегрується з іншими технологіями та системами. Наприклад, робот може бути підключений до системи автоматизації будинку, такої як Home Assistant. В такому випадку робот може отримувати команди від інших пристроїв у будинку та реагувати на події, що відбуваються в системі. Це дає змогу створювати складні автоматизовані сценарії, наприклад, коли робот починає прибирання в певний час або реагує на те, що в будинку немає людей.

Ще однією важливою перевагою Open Interface є можливість для реального часу моніторингу роботи робота. Користувач може отримувати дані про місцезнаходження робота, стан сенсорів, статус батареї, а також зчитувати важливу інформацію про зовнішні фактори, як перешкоди на шляху чи ступінь запиленості. Такі можливості відкривають двері для створення складних алгоритмів, наприклад, для адаптивного маршруту прибирання, де робот змінює свою стратегію в залежності від типу приміщення або умов навколишнього середовища.

Open Interface має також велику підтримку з боку спільноти розробників, що робить його ще більш доступним для використання. Завдяки відкритій документації і специфікаціям, програмісти можуть легко освоїти цей стандарт і створювати власні рішення, адаптуючи робота до своїх потреб. Більш того, велика кількість готових прикладів і кодів дає можливість заощадити час на розробку нових функцій, зосередивши увагу на креативному застосуванні технологій.

Цікаво, що Open Interface також використовується в навчальних цілях. Завдяки його простоті та доступності, студенти та дослідники можуть вивчати основи робототехніки і програмування, використовуючи робота як платформу для експериментів і досліджень. Такий підхід дозволяє на практиці ознайомитися з принципами роботи автономних систем, а також дає можливість створювати власні проекти та рішення, що можуть бути використані в реальному житті.

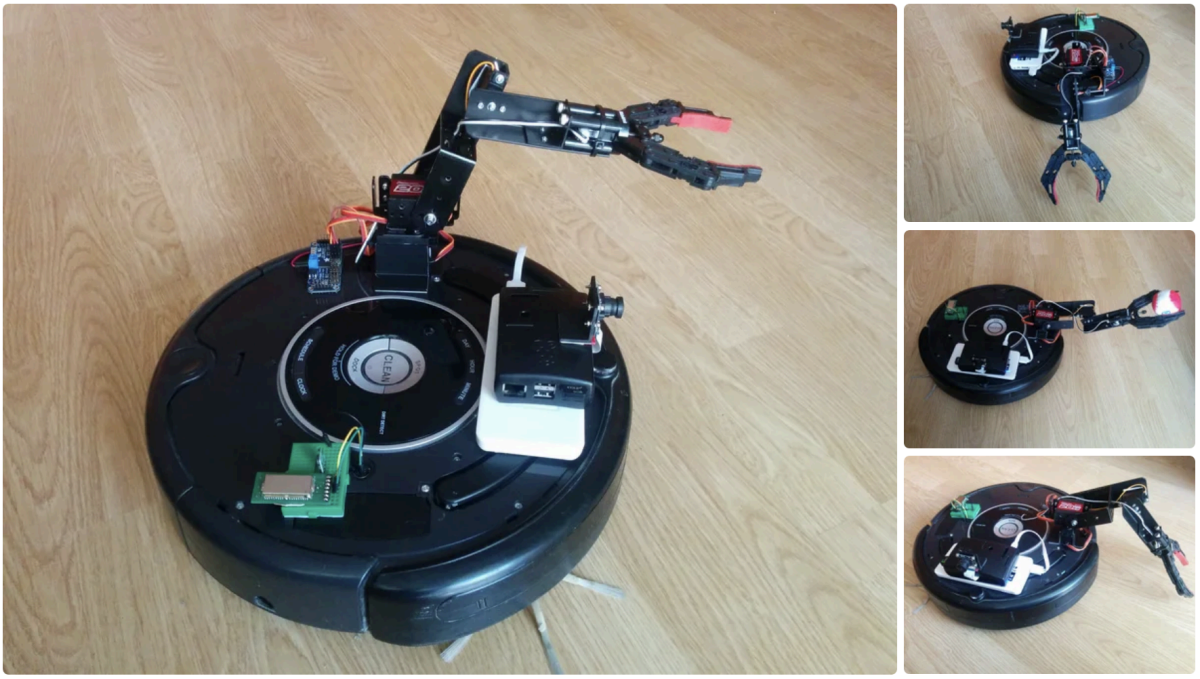


Рисунок 2.12 Приклад керування додатковими компонентами робота Roomba

Не менш важливою є можливість підключення додаткових сенсорів і модулів до робота через Open Interface. Це дозволяє розширювати функціональність пристрою, додаючи, наприклад, камери, лазерні далекоміри, датчики температури чи навіть нові типи сенсорів, що збільшують здатність робота орієнтуватися в середовищі та виконувати більш складні завдання.

Завдяки всім цим можливостям Open Interface є надзвичайно потужним інструментом для створення інноваційних рішень у галузі робототехніки. Він дозволяє створювати мобільних роботів з унікальними властивостями, адаптованими до різних умов, та відкриває нові горизонти для розвитку автономних систем і розумних будинків.

3 ОЗВУЧЕННЯ ТА ОБРОБКА ГОЛОСОВОГО ПАКЕТУ

Одним із найцікавіших і водночас найвідповідальніших етапів у процесі створення власного голосового пакету для робота-пилососа є озвучення. Саме завдяки голосу робот спілкується з користувачем, реагує на команди, повідомляє про події та створює загальне враження про себе. Голос може бути простим і нейтральним, а може мати характер, стиль і навіть власну «особистість». Коли я починав роботу над цим проєктом, стало очевидно, що більшість доступних моделей роботів взагалі не мають україномовної озвучки. Там, де вона є — вона часто звучить неприродно: або з акцентом, або як машинне читання без емоцій, згенероване стандартним Google Text to Speech. Саме тому я вирішив створити перший український кастомний голосовий пакет — якісний, унікальний і такий, що залишає приємне враження під час щоденного користування.

Для свого голосового пакету я обрав образ GLaDOS — штучного інтелекту з культової гри Portal 2. Вона має специфічний саркастичний, трохи зверхній стиль, і водночас говорить надзвичайно чітко та виразно. Завдання полягало в тому, щоб передати ці риси українською мовою, у форматі, придатному для використання в домашньому пристрої. Озвучка такого персонажа вимагає тонкої роботи — як з технічного, так і з художнього боку.

Є два основні способи, якими можна створити голос для робота. Перший — це синтетична генерація голосу за допомогою сучасних голосових моделей. Наприклад, один із найпопулярніших інструментів — це Eleven Labs. Цей сервіс дає можливість вводити текст і отримувати в результаті звучання, яке дуже близьке до природного людського голосу. Платформа також дозволяє створювати власні голоси — або на основі вже готових профілів, або натренованих на кількох хвилинах прикладного аудіо. Такий підхід значно пришвидшує процес озвучення, адже не потрібно залучати акторів, організовувати запис і обробку. До того ж, за допомогою Eleven Labs можна створити озвучку не лише «нейтральну», а й стилізовану — з відповідною емоцією, інтонацією, паузами.

Другий підхід — це запис живого голосу з подальшим редагуванням. І саме його я обрав для реалізації проєкту. Було важливо не просто отримати україномовний голос, а досягти емоційного впливу — відчути характер GLaDOS у кожній фразі. Для цього я звернувся до професійної акторки дубляжу, яка свого часу озвучувала Portal 2 українською. Саме її голос став основою для пакету. Перевага такого підходу полягає у високій автентичності. Живий голос має нюанси, які складно змоделювати: ледь помітні інтонації, зміну ритму, природне звучання пауз. Усе це допомагає створити справжнє враження того, що робот не просто читає рядки коду, а дійсно «говорить». [37]

Після запису кожна репліка проходила обробку. Спершу — технічну: очищення шумів, нормалізація гучності, вирізання зайвих пауз. Далі — художню: було додано електронні ефекти, які надали голосу легкого «роботичного» звучання. Це дозволило досягти балансу між людською природністю і штучною атмосферністю, характерною для голосу GLaDOS. У результаті фрази звучать чітко, трохи холодно, з впізнаваним іронічним тоном — саме те, що було потрібно.

Цікаво, що навіть при використанні голосу жінки, такий стиль може бути створений і на базі чоловічого голосу. Усе залежить не стільки від тембру, скільки від інтерпретації. Але для повної відповідності персонажу було вирішено працювати саме з жіночим голосом. [38]

Якщо порівнювати обидва підходи — синтетичний і живий — кожен має свої переваги. Синтетичний дає змогу швидко протестувати ідеї, легко змінювати фрази, не витрачаючи час на повторний запис. Але при цьому якість та емоційна глибина ще поки відчутно поступаються живій озвучці. Особливо це помітно при спробі передати гумор, іронію або характер персонажа. Живий голос, натомість, потребує більше часу, ресурсів та технічних навичок для обробки, але результат виходить глибший, цікавіший і приємніший у щоденному використанні.

Саме завдяки цьому підходу мені вдалося створити повноцінний голосовий образ для робота-пилососа, який тепер спілкується українською не просто інформативно, а цікаво й атмосферно. І хоча на перший погляд це лише набір

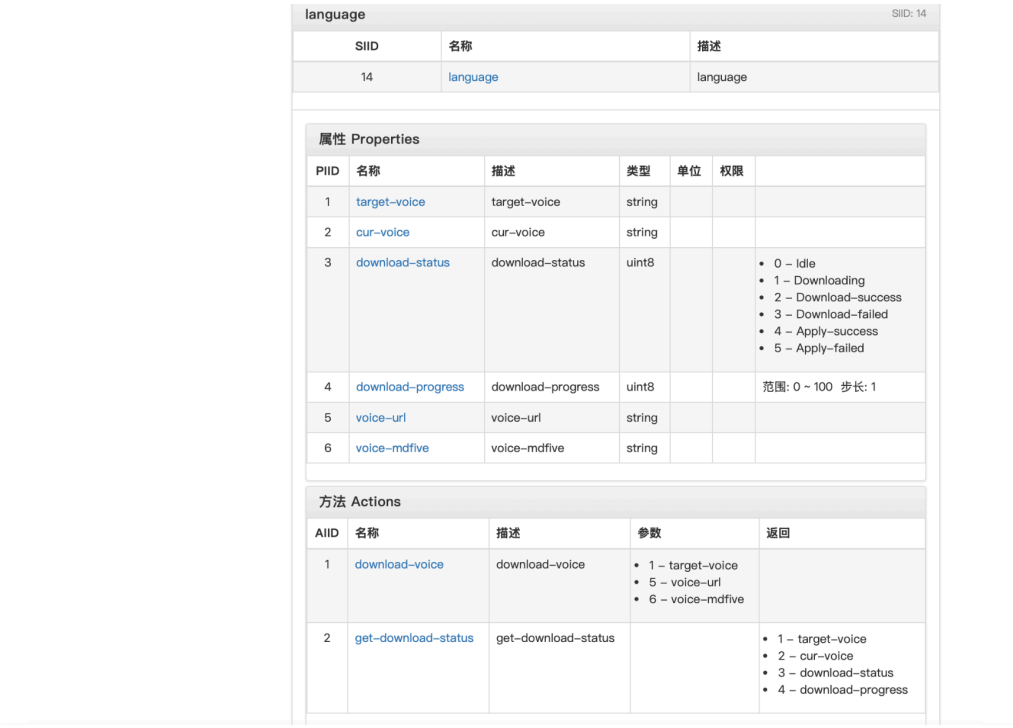
голосових повідомлень, на практиці він формує абсолютно нове враження від техніки. Звичайний пристрій починає виглядати як частина особистого простору, як щось ближче до живого помічника, а не просто електронного гаджета.

3.1 Робота з програмою Python-miio

Щоб взаємодіяти з роботом-пилососом напряму, минаючи офіційні додатки, було створено проєкт python-miio. Це відкритий проєкт із відкритим вихідним кодом, який реалізує протокол обміну даними між пристроями Xiaomi та користувачем. Завдяки python-miio можна надсилати команди пристроям, змінювати їх налаштування і навіть завантажувати нові голосові пакети.

Протокол Xiaomi miio — це внутрішній спосіб, яким пристрої бренду Xiaomi обмінюються інформацією через локальну мережу. Він описує, як саме має виглядати запит, які параметри треба вказати, і як виглядає відповідь пристрою. Оскільки виробник офіційно не надає повної документації, спільнота користувачів провела зворотну інженерію, поступово відновлюючи специфікацію протоколу. [39]

Важливим джерелом інформації для роботи з пристроями є сайт Miot Specification (доступний за посиланням home.miot-spec.com). Там можна знайти офіційно описані сервіси і властивості для величезної кількості пристроїв Xiaomi та партнерських брендів. Наприклад, там можна дізнатися, який саме ідентифікатор має властивість голосового пакету або яка команда відповідає за запуск прибирання.



language SIID: 14

SIID	名称	描述
14	language	language

属性 Properties

PIID	名称	描述	类型	单位	权限	
1	target-voice	target-voice	string			
2	cur-voice	cur-voice	string			
3	download-status	download-status	uint8			• 0 – Idle • 1 – Downloading • 2 – Download-success • 3 – Download-failed • 4 – Apply-success • 5 – Apply-failed
4	download-progress	download-progress	uint8			范围: 0 ~ 100 步长: 1
5	voice-url	voice-url	string			
6	voice-mdfive	voice-mdfive	string			

方法 Actions

AIID	名称	描述	参数	返回
1	download-voice	download-voice	• 1 – target-voice • 5 – voice-url • 6 – voice-mdfive	
2	get-download-status	get-download-status		• 1 – target-voice • 2 – cur-voice • 3 – download-status • 4 – download-progress

Рисунок 3.1 Відображення специфікації протоколу Xiaomi Mio

У випадку з роботами-пилососами, python-miio дозволяє виконувати “сирі” команди через локальну мережу. Це дає можливість, наприклад, вказати роботу завантажити інший голосовий пакет, навіть якщо виробник це офіційно не передбачив. Для цього потрібно знати IP-адресу пристрою, його унікальний токен безпеки, а також точні параметри команди.

Такий підхід відкриває величезні можливості для гнучкого налаштування розумних пристроїв, хоча і потребує певної обережності, адже неправильна команда може призвести до помилок у роботі пристрою. [40]

3.2 Мапінг голосових команд та адаптація озвучки

Коли йдеться про створення повноцінного україномовного голосового пакету, важливу роль відіграє не лише саме озвучення, а й правильне структурування та редагування фраз, які відтворює робот. Мова не про просте перекладання слів — мова про те, як зробити голос не лише зрозумілим, а ще й цікавим, впізнаваним і стильовим. Саме тому наступним етапом після створення аудіофайлів стало завдання мапінгу — тобто відповідності кожної фрази з голосового пакету конкретній функції пристрою.

Починається все з того, що потрібно отримати оригінальний голосовий пакет, який використовується у прошивці робота-пилососа. У більшості випадків він представлений у вигляді архіву або набору окремих аудіофайлів, де кожна фраза збережена у певному форматі, наприклад .wav, .mp3 або .pcm. Ці файли зазвичай мають стандартні назви на кшталт clean_start.wav, dock.wav, error_01.wav — тобто назви, які вказують на функцію або подію, з якою вони пов’язані.

На цьому етапі я почав роботу з англомовними фразами, які були стандартними для конкретної моделі робота. Ці фрази доволі лаконічні: “Starting cleaning”, “Going to dock”, “Error: Cliff sensor”. І хоча функціонально вони виконують своє завдання, вони звучать надто формально і беземоційно. Саме тому було вирішено повністю переписати їх — спочатку перекласти українською, а далі адаптувати до стилістики GLaDOS із Portal. [41]

	A	B	C	D	E
1	1	**back_dock_failed.wav**	**Couldn't reach the dock. Please remove obstacles and make sure the charging surfaces are clean!**	**Не вдалося дістатися до док-станції. Приберіть перешкоди та очистіть зарядні контакти!**	**Не вдається дістатись док-станції. Приберіть це сміття навколо мене та очистіть зарядні контакти. Це місце справжнє звалище**
2	2	back_dock_nearby.wav	Could not reach docking station, please remove obstacles!	Не вдалося дістатися до док-станції, будь ласка, приберіть перешкоди!	Не вдається дістатися док-станції. Приберіть це сміття з мого шляху. Це місце справжнє звалище.
3	3	bin_in.wav	Dustbin installed.	Пилозбірник встановлено.	Пилозбірник встановлено
4	4	bin_out.wav	Dustbin removed.	Пилозбірник знято.	Пилозбірник знято
5	5	binout_error10.wav	Filter is clogged. Remove the waste bin and clean the filter!	Фільтр засмічений. Зніміть пилозбірник та очистіть фільтр!	42.ogg У мене надто брудний фільтр, очистьте його щоб я продовжила роботу
6	6	bl_recovery_bootfailed.wav	Bootfailure: It will take about 5 minutes to start restoring the initial version. Please be patient.	Збій завантаження: Відновлення початкової версії займе близько 5 хвилин. Зачекайте будь ласка!	Збій завантаження: Відновлення початкової версії системи займе близько п'яти хвилин.
7	7	bl_recovery_failed.wav	Restore the initial version failed. Please try again.	Не вдалося відновити початкову версію. Спробуйте ще раз.	Не вдалося відновити початкову версію системи. Спробуйте ще раз.
8	8	bl_recovery_retry.wav	Restart the initial version. It will take about 5 minutes. Please be patient.	Перезапуск початкової версії. Це займе близько 5 хвилин. Зачекайте, будь ласка.	Перезапуск відновлення початкової версії системи. Це займе близько п'яти хвилин
9	9	bl_recovery_start.wav	Restoring the initial version. It will take about 5 minutes. Please be patient.	Відновлення початкової версії. Це займе близько 5 хвилин.	Відновлюю початковий стан системи. Це займе близько п'яти хвилин
10	10	bl_recovery_updatefailed.wav	The upgrade failed. It will take about 5 minutes to start restoring the initial version. Please be patient.	Не вдалося виконати оновлення. Відновлення початкової версії. Це займе близько 5 хвилин	Не вдалося виконати оновлення. Відновлюю початкову версію системи. Це займе близько п'яти хвилин.
11	11	charging.wav	Charging.	Заряджання.	47.ogg Я відчуваю приплив сил.
12	12	clean_bin.wav	Please clean the dustbin.	Будь ласка, очистіть пилозбірник.	Схоже, пилозбірник досяг критичної маси. Очистіть його.

Рисунок 3.2 Приклад мапінгу голосових команд робота Roborock в стилі персонажа відеогри Portal

Переклад — це не механічна операція, а живий творчий процес. Наприклад, замість “Starting cleaning” простою мовою можна сказати “Починаю прибирання”. Але якщо ми хочемо передати характер GLaDOS, то ця ж фраза перетворюється на щось на кшталт: “Чудово. Мені знову доведеться все робити самій.” Такий підхід створює зовсім інший настрій: фраза залишається функціональною, але тепер вона ще й має характер, додає особистості й гумору. Подібні варіанти не просто

інформують користувача, а викликають усмішку, створюють атмосферу гри, що додає побутовому пристрою несподіваної глибини.

У процесі адаптації кожної фрази я дотримувався балансу між трьома речами: інформативністю (щоб було зрозуміло, що відбувається), українською мовною нормою (щоб не втратити природність мови), та стилістикою GLaDOS (щоб не загубився той характерний іронічний підхід, за який ми всі її любимо). Деякі фрази доводилося переформульовувати повністю — особливо ті, які у оригіналі звучали занадто технічно або мали складну термінологію. Наприклад, “Error: brush motor” перетворювалось на щось на кшталт “О, так. Знову щось зламалося. Як несподівано.”

Кожна така адаптація проходила кілька етапів. Спочатку створювався чорновий переклад. Потім — редагування з урахуванням стилю. Далі — перевірка, чи не виходить фраза за часові межі, які підтримує прошивка робота. І лише після цього — озвучення і фінальна обробка. Часто доводилося вносити правки прямо під час запису, щоби зберегти правильну інтонацію й темп.

Цей підхід дозволяє не просто створити україномовну локалізацію, а повноцінно кастомізувати характер робота. Голосовий мапінг із такою адаптацією — це той самий інструмент, який перетворює робот-пилосос із мовчазного інструмента на співрозмовника з індивідуальністю. І саме це, як на мене, є основною цінністю кастомного голосового пакету: він не просто говорить українською — він говорить цікаво, стильно, влучно.

3.3 Створення голосового пакету для робота-пилососа Roborock

Одним із ефективних способів кастомізації функціоналу роботів-пилососів є модифікація їхнього голосового супроводу. Зокрема, для пристроїв Roborock першого і другого покоління (таких як Roborock v1, M1S, S4, S5, S5e, S6, S6 MaxV, S6 Pure, T6 передбачена можливість встановлення власних голосових пакетів. Це дозволяє замінити стандартні голосові повідомлення на індивідуально створені аудіофайли або на основі інших джерел. [42]

Процес встановлення базується на використанні відкритих програмних інструментів. У репозиторії glados_ukr представлений україномовний голосовий пакет, створений за мотивами персонажа GLaDOS із гри Portal. Для початку необхідно завантажити файл roborock_glados.pkg на комп'ютер. Далі потрібно встановити утиліту python-miio, що дозволяє взаємодіяти з пристроєм через локальну мережу.

У каталозі з завантаженим файлом необхідно відкрити командний рядок та виконати команду:

```
mirobo --ip 192.168.0.124 --token
50597y49p1894f6d6e67477349747408 install-sound roborock_glados.pkg
```

Параметри IP-адреси та токена повинні відповідати конкретному пристрою. Токен можна отримати за допомогою команди miio, наприклад python -m miio.cli cloud list для Windows або miiocli cloud list для Linux та MacOS.

Важливо враховувати, що для роботів Roborock третього покоління встановлення власних голосових пакетів є неможливим через використання сертифікатів підпису файлів, які перевіряються пристроєм перед оновленням. Процес створення власного голосового пакету також є доступним для реалізації. Перш за все, необхідно підготувати аудіофайли у форматі .wav, дотримуючись наступних технічних характеристик: бітрейт 1411,2 кбіт/с, частота дискретизації 44,1 кГц або 16 кГц. У разі наявності аудіофайлів у форматі .mp3 або іншому форматі конвертацію у потрібний формат зручно здійснювати за допомогою утиліти ffmpeg. Для перетворення файлу voice.mp3 у формат .wav із частотою дискретизації 44,1 кГц можна використати команду:

```
ffmpeg -i voice.mp3 -ar 44100 -ac 1 -sample_fmt s16 voice.wav
```

Тут параметр -ar 44100 визначає частоту дискретизації 44,1 кГц, -ac 1 вказує на використання одного аудіоканалу (моно), а -sample_fmt s16 встановлює 16-бітну глибину вибірки.

Після підготовки усіх необхідних аудіофайлів у форматі .wav потрібно заархівувати їх та зашифрувати для створення пакету встановлення. Для цього

використовується утиліта ccrypt. Перебуваючи у каталозі з файлами, необхідно виконати команду:

```
tar zc *.wav | ccrypt -e -K "r0ckrobo#23456" > roborock_myvoice.pkg
```

Ця команда виконує архівацію усіх .wav файлів у стиснений потік за допомогою утиліти tar, після чого відразу передає цей потік на шифрування через ccrypt, використовуючи стандартний ключ "r0ckrobo#23456". У результаті створюється файл із розширенням .pkg, який готовий для залиття на пристрій.

Після створення голосового пакету його можна залити на робот-пилосос, використовуючи команду, аналогічну процедурі встановлення готового пакету.

Таким чином, створення власного голосового пакету для роботів Roborock є доступною процедурою, що базується на використанні відкритих програмних інструментів і дозволяє значно розширити можливості персоналізації пристрою. [43]

3.4 Створення голосового пакету для робота-пилососа Dreame

Після успіху кастомного озвучення для Roborock багато ентузіастів вирішили спробувати щось подібне і для пилососів Dreame. Dreame, якщо коротко, — це один із суббрендів Xiaomi, який спеціалізується на розробці інтелектуальної побутової техніки, зокрема роботів-пилососів, які вирізняються сучасним дизайном і хорошою функціональністю. Як і у випадку з Roborock, ідея полягала в тому, щоб замінити стандартний голос пристрою на щось більш емоційне й особисте — наприклад, голос відомого персонажа чи навіть власний.

Для початку варто було обрати спосіб встановлення нового голосового пакету. Один з варіантів робота через Home Assistant — популярну платформу для керування розумним будинком. Home Assistant дозволяє об’єднати в єдину систему величезну кількість різноманітних пристроїв — від лампочок до роботів-прибиральників — і налаштувати автоматизації за власними правилами.

Щоб змінити голос Dreame через Home Assistant, спершу потрібно встановити відповідну інтеграцію. Якщо йдеться про Dreame Vacuum, інтеграцію

легко додати через HACS — це спеціальний каталог сторонніх доповнень для Home Assistant. Якщо коротко, HACS працює як “магазин додатків”, але для компонентів і розширень, яких немає у стандартній поставці Home Assistant. Завдяки цьому HACS спрощує встановлення і оновлення різноманітних інтеграцій.

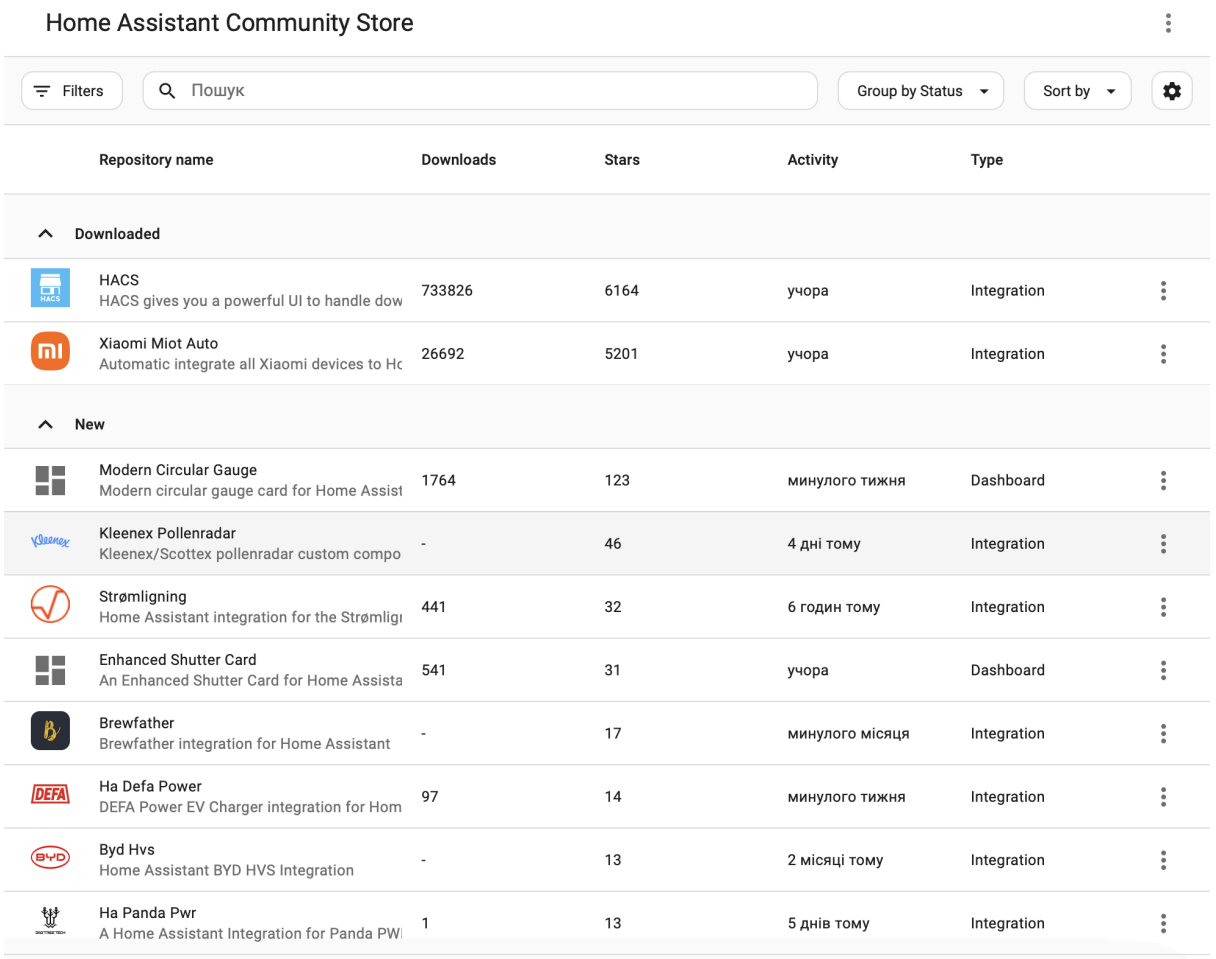


Рисунок 3.3 Home Assistant Community Store

Після встановлення інтеграції користувач переходить до розділу “Інструменти для розробників” у Home Assistant, де можна вручну запускати різні сервіси. [44]

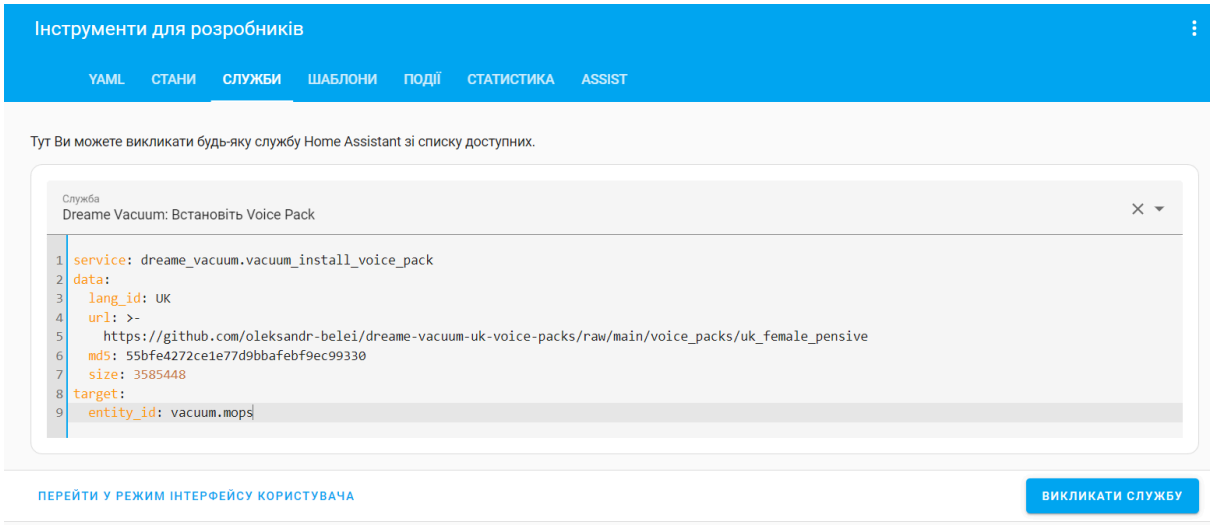


Рисунок 3.4 Робота з інтеграцією Dreame Vacuum в інтерфейсі Home Assistant

Там потрібно було переключитися в режим YAML — це формат для опису даних, який широко використовують у світі автоматизації. У YAML вставлявся спеціальний фрагмент коду, де задавалися посилання на архів із голосовим пакетом, його контрольна сума md5, розмір у байтах та мова озвучення. Дуже важливо було замінити ідентифікатор пристрою (entity_id) на правильний — той, який відповідає саме вашому пилососу. Після чого залишалося тільки натиснути кнопку для виконання дії — і робот отримував новий голос.

Інший варіант передбачав використання інтеграції Xiaomi Miot Auto, яка відкриває доступ до широкого списку можливостей для пристроїв екосистеми Xiaomi.

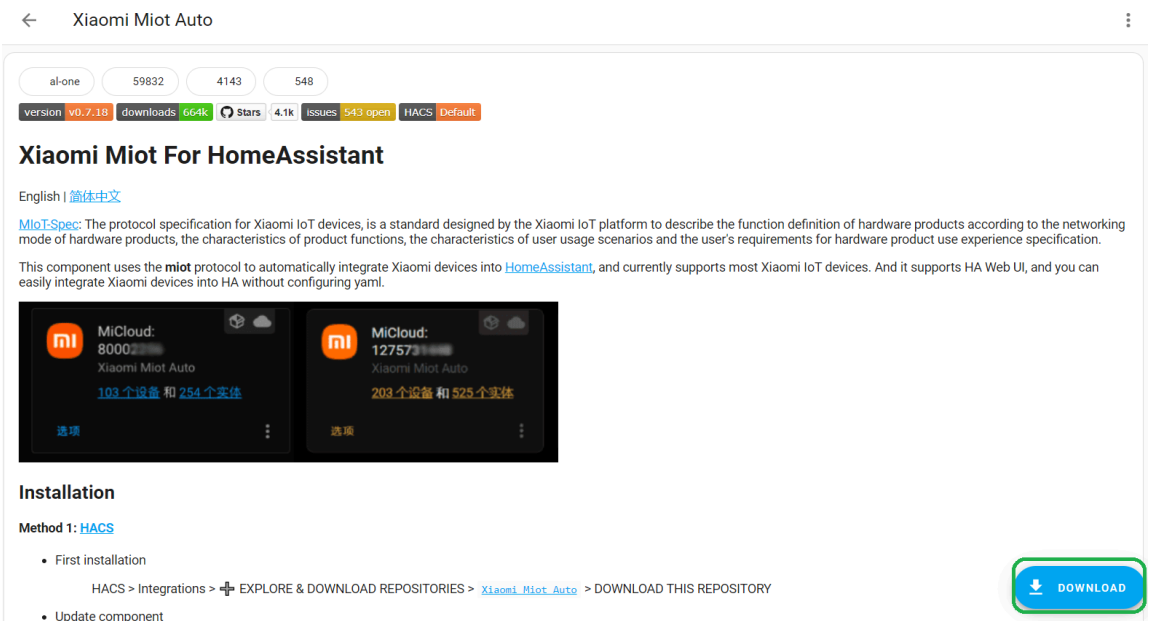


Рисунок 3.5 Інтеграція Xiaomi Miot для Home Assistant

Принцип роботи залишався схожим, проте команди для зміни голосу подавалися через інший сервіс і з трохи іншими параметрами — потрібно було вказати спеціальні внутрішні ідентифікатори сервісів і властивостей, такі як siid і piid.

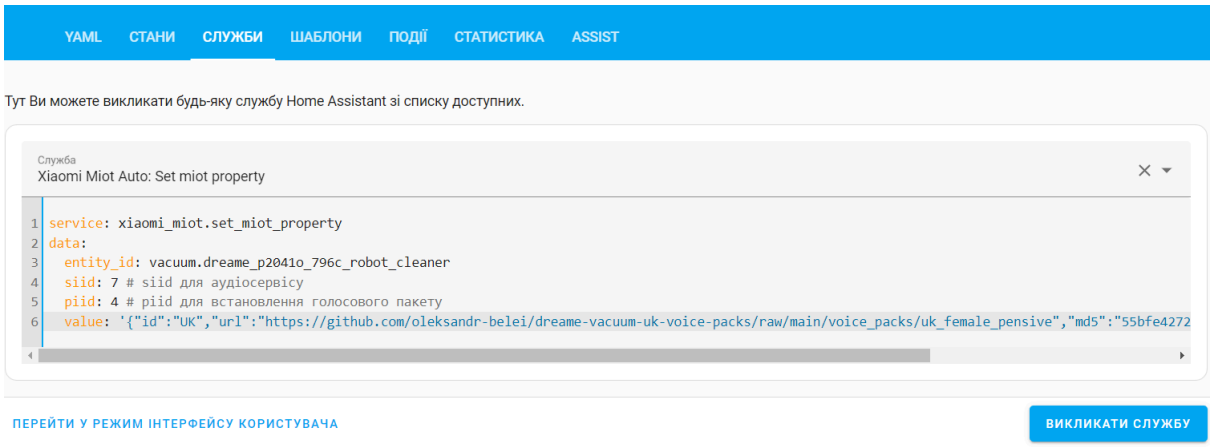


Рисунок 3.6 Виконання команди за допомогою інтеграції Xiaomi Miot

Більш просунуті користувачі могли піти ще далі та використовувати інструмент python-miio. Це консольна утиліта на Python, яка дозволяє напряму спілкуватися з пристроями Xiaomi через мережу. Це опенсорсний інструмент, який намагається повторити роботу протоколу miio. Він дозволяє напряму спілкуватися з пристроями Xiaomi через консоль без потреби у фірмових додатках або сервісах. Саме завдяки python-miio багато ентузіастів змогли експериментувати з голосами, перепрошивками та розширеними налаштуваннями своїх пристроїв.

За допомогою python-miio можна надсилати команди для зміни голосового пакету без посередництва Home Assistant. Тут потрібно знати IP-адресу пилососа у локальній мережі та його токен доступу — спеціальний ключ, який дає змогу управляти пристроєм. Передача даних також здійснювалася через формат сирих команд, де вказувався потрібний URL на архів із голосовим пакетом, контрольна сума й розмір файлу.

```
python -m miio.cli genericmiot --ip 192.168.50.157 --
token 614a498f6c72506d6e3066764f73696a raw_command set_properties "
[{'did': '8023334994', 'siid': 7, 'piid': 4, 'value': '{\"id\":\"UK\",\"url\":\"https://awsde0.fds.api.xiaomi.c
om/dreame-
product/resources/59bcaf5201f15950c12917d0bb505321\",\"md5\":\"59bcaf5201f15950c12917d0bb505321\",\"size\":3927
702}'}]"
```

Рисунок 3.7 викорнання команди за допомогою python-miio

Ще однією цікавою альтернативою було використання Valetudo. Це незалежна прошивка для роботів-пилососів, яка повністю замінює фірмові сервери на локальне управління. Якщо ваш Dreame працював під Valetudo, процес встановлення нового голосового пакету спрощувався до елементарного: достатньо було зайти у веб-інтерфейс пристрою через браузер, знайти розділ “Налаштування робота”, у підрозділі “Інші налаштування” заповнити параметри нового голосу й натиснути кнопку для збереження змін. [45]

Голосовий пакет містить архів з файлами фраз у форматі .ogg із частотою 16 000 Гц. Голосовий пакет для робота зазвичай є архівом у форматі tar.gz. Якщо файл має тільки розширення .gz або взагалі не має розширення, його можна вручну перейменувати, додавши .tar.gz в кінці назви. Після цього архів легко відкриється будь-якою програмою для роботи з такими файлами, що дозволить переглядати і редагувати вміст пакету.

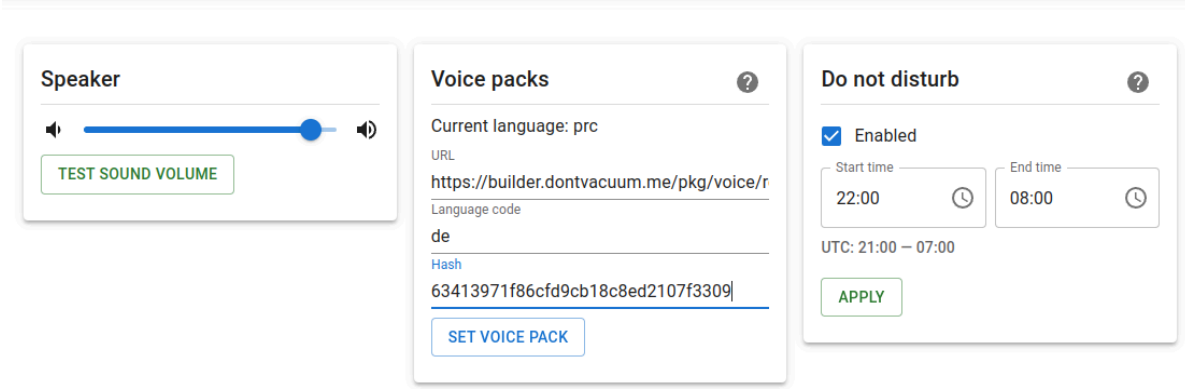


Рисунок 3.8 Завантаження голосового пакету за допомогою інтерфейсу Valetudo

Якщо користувач створював власний пакет, наприклад записуючи свій голос або генеруючи його за допомогою синтезатора мови, потрібно було дотримуватися чіткої відповідності назв файлів і змісту, оскільки інакше робот міг некоректно їх відтворювати. Усі фрази зберігалися в одному архіві без вкладених папок.

Також потрібно було завантажити готовий архів в інтернет, щоб надати пристрою пряме посилання. Для цього підходили такі сервіси, як GitHub, GitLab або навіть звичайні FTP-сервери. Щоб визначити розмір архіву в байтах і

розрахувати md5-хеш, можна було скористатися простими онлайн-інструментами або локальними утилітами, якщо користувач мав трохи більше технічного досвіду.

У підсумку цей процес відкривав безліч можливостей для творчості: від створення власного озвучення до адаптації улюблених персонажів або мемів. Багато користувачів захопилися цією ідеєю настільки, що створювали цілі тематичні пакети: наприклад, голоси героїв мультфільмів, саркастичні версії із гумористичним текстом або навіть озвучення під класичні фільми.

3.5 Розробка та встановлення голосового пакету для робота-пилососа Midea M7 Pro

Індивідуалізація голосового супроводу в роботах-пилососах стає дедалі популярнішою серед користувачів, які прагнуть надати пристроям характеру, а також інтегрувати їх у локалізоване середовище. Midea M7 Pro підтримує функцію кастомізації звукових повідомлень, що дає змогу замінити стандартні голосові пакети на індивідуальні файли користувача. Цей процес не є надто складним, однак вимагає базового розуміння роботи з Android Debug Bridge (ADB) та дотримання чіткої структури файлів.

Для початку необхідно підготувати середовище на персональному комп’ютері. Першим кроком є завантаження пакету SDK Platform Tools, який містить утиліту ADB. Цей набір інструментів є офіційним від Google і широко використовується для взаємодії з Android-пристроями через командний рядок. Після завантаження архів слід розпакувати у зручну директорію, наприклад, C:\platform-tools.

Після цього потрібно фізично з’єднати робота з комп’ютером. USB-порт на моделі Midea M7 Pro розташований під верхньою кришкою пристрою, поруч із кнопкою скидання (Reset). Він зазвичай закритий декоративною заглушкою, яку слід обережно зняти. Для передачі даних використовується звичайний USB-кабель, однак варто звернути увагу, що не всі кабелі підходять — деякі з них підтримують лише зарядку без передачі даних. У разі, якщо команда adb devices не повертає ідентифікатор пристрою, варто спробувати інший кабель.

Після успішного підключення та розпізнавання робота можна переходити до підготовки голосового пакету. Структура звукових файлів повинна відповідати певним технічним вимогам: формат mp3, частота дискретизації 48000 Hz, бітрейт 64 kb/s, моно. Важливо дотримуватись саме цих параметрів, інакше пристрій не зможе відтворити файл або він буде звучати некоректно. Для цього зручно використовувати аудіоредактори на зразок Audacity, які дозволяють експортувати аудіо з необхідними параметрами.

Після підготовки звуків, їх слід помістити у зручну директорію, наприклад C:\midea-sounds. Далі необхідно створити BAT-файл — це текстовий скрипт, що автоматизує процес завантаження файлів на внутрішню файлову систему робота. Скрипт здійснює рекурсивний обхід усіх підкаталогів у заданій директорії та копіює кожен файл у відповідне місце /oem/sound/ на пристрої. Такий підхід дозволяє підтримувати структуру каталогів та, за потреби, додавати окремі звукові варіанти для різних ситуацій або мовних локалей. Вміст BAT-файлу виглядає наступним чином:

```
@echo off
set SOURCE_FOLDER=повний_шлях_до_текі_зі_звуками
for /r "%SOURCE_FOLDER%" %%x in (*) do (
  set "file=%%x"
  setlocal enabledelayedexpansion
  set "path=!file:%SOURCE_FOLDER%\=!"
  adb push "%%x" "/oem/sound/!path!"
  endlocal
)
```

Після збереження цього файлу із розширенням .bat (наприклад, upload-voicepack.bat), його можна запустити подвійним кліком або через термінал у тій самій теці, де розташовані adb.exe та аудіофайли. У процесі виконання скрипт завантажить усі звуки на робот, після чого вони автоматично використовуватимуться замість стандартних повідомлень. Зазвичай

перезавантаження пристрою не вимагається, однак у разі відсутності змін його варто виконати вручну.

Цей метод не потребує змін прошивки робота або root-доступу, однак потенційно може вплинути на гарантійні зобов’язання, оскільки втручання у файлову систему не передбачене звичайним користувацьким сценарієм. Водночас, відкритість операційної системи Midea дає змогу реалізовувати такі кастомізації доволі просто та без ризику “заблокувати” пристрій.

3.6 Python аудіо генератор

Репозиторій Dustcloud в розділі audio_generator для пристроїв xiaomi.vacuum спеціалізується на створенні голосових фраз (команд) для роботів-пилососів, зокрема для моделей Roborock. Цей інструмент дозволяє генерувати голосові повідомлення, які робот озвучує під час виконання різних операцій. Генерація голосу здійснюється через використання CSV файлу, що містить текстові фрази, які будуть перетворені на мову. Користувач має можливість не лише змінювати текстові фрази для команди, але й переписати сам CSV файл під свої індивідуальні потреби, що дає значну гнучкість у налаштуванні голосових повідомлень.

Процес генерації озвучених файлів здійснюється за допомогою різних рушіїв TTS (Text-to-Speech), таких як:

gTTS (Google Text-to-Speech): популярний сервіс від Google, який використовує технології глибокого навчання для перетворення тексту на мову.

eSpeak NG: відкрите програмне забезпечення для синтезу мови, що підтримує велику кількість мов і акцентів.

macOS TTS: інтегрований інструмент для перетворення тексту на мову в операційних системах Mac OS X, з можливістю вибору різних голосів безпосередньо з системи.

Amazon Polly: хмарний сервіс від Amazon, що забезпечує високу якість синтезу мови і підтримує численні мови та голоси.

Після того, як текстові фрази будуть перетворені в звукові файли за допомогою одного з цих рушіїв, інструмент створює спеціальний пакет в форматі .pkg. Цей пакет є сумісним з пристроями Roborock, що дозволяє зручно інсталювати нові голосові команди на роботі-пилососи, аби вони озвучували повідомлення під час виконання різних завдань. Це може бути особливо корисно для створення персоналізованих інтерфейсів, де кожен робот буде взаємодіяти з користувачем на своєму «особистому» мовному рівні.

```
Available localized audio instructions:
1. language/audio_it.csv
2. language/audio_de.csv
3. language/audio_fi.csv
4. language/audio_es.csv
5. language/audio_ca.csv
6. language/audio_uk.csv
7. language/audio_cs.csv
8. language/audio_no.csv
9. language/audio_nl.csv
10. language/audio_tr.csv
11. language/audio_ru.csv
12. language/audio_fr.csv
13. language/audio_se.csv
14. language/audio_pl.csv
15. language/audio_en.csv
Please select option by typing number (1-15): 6
Available TTS engines:
1. gtts
2. macos
Please select option by typing number (1-2): 1
back_dock_nearby.wav
File 'generated_uk/back_dock_nearby.wav' already exists. Overwrite? [y/N] y
bin_in.wav
File 'generated_uk/bin_in.wav' already exists. Overwrite? [y/N]
```

Рисунок 3.9 Робота python застосунку audio generator від Dustcloud

Цей процес відкриває можливості для адаптації голосових команд під різні мови, акценти та навіть стиль озвучення, даючи користувачам можливість створювати інтерфейси, що ідеально підходять для їхніх конкретних потреб. За допомогою цього інструменту можна не лише змінювати стандартні голосові фрази, але й розширювати функціональність робота-пилососа, роблячи його більш зручним і персоналізованим для щоденного використання.

3.7 Реалізація зберігання голосових пакетів

Для забезпечення уніфікованого та масштабованого зберігання озвучок на серверній частині платформи пропонується централізована база метаданих, яка

описує кожен голосовий пакет незалежно від його внутрішнього формату. Кожна озвучка розглядається як логічна одиниця з унікальним ідентифікатором, назвою, мовою, стилем, автором (опційно), та набором посилань на фізичні файли, призначені для конкретних моделей роботів. Наприклад, один і той самий голосовий пакет може містити посилання на .pkg-архів для Roborock, .tar.gz з .mp3 для Xiaomi, .tar.gz з .ogg для Dreame. Ці варіації мають однакову смислову основу, однак адаптовані до технічних обмежень кожної лінійки пристроїв. [46]

Кожна варіація зберігається з прив'язкою до моделі, наприклад xiaomi.vacuum.b108gl, а також включає обов'язкові метадані: пряме посилання (url) на файл, контрольну суму (md5) для перевірки цілісності пакету, розмір файлу в байтах (size). Всі ці параметри є критично важливими під час завантаження озвучки на робот, оскільки протокол MIOT для багатьох моделей передбачає передачу цих значень у вигляді структурованої команди.

Для прикладу, згідно з документацією та спостереженнями користувачів python-miio, завантаження озвучки в окремих моделях, таких як dreame.vacuum.r2294s чи xiaomi.vacuum.b108gl, здійснюється через raw_command типу action, в якому необхідно передати did пристрою, siid, aiid, а також масив in з параметрами — мовним кодом, URL архіву, його MD5, та розміром. Приклад такої команди:

```
        {
  "id":12345,
  "method":"action",
  "params":{
    "did":"DID",
    "siid":24,
    "aiid":2,
    "in":[
      {"piid":3, "value":"DE"},
      {"piid":4,
```

"value":"https://awsde0.fds.api.xiaomi.com/dreame-product/dreame.vacuum.p2008/voices/package/deyu.tar.gz"},
{"piid":5, "value":"96ab843008bb4d19480ac8a07d648aa7"},
{"piid":6, "value":1391385}
]
}
}

У даному випадку всі параметри мають бути достовірними, інакше робот відповість помилкою (code: -1). Це вимагає, щоби кожен голосовий пакет у базі платформи мав достовірну та перевірену інформацію.

Ще одним джерелом для формування або верифікації списку доступних озвучок є файли soundpackage.json, які можна знайти за кількома URL-адресами, що відповідають різним піддоменам Xiaomi API. Такі файли містять службову інформацію про всі офіційні озвучки для конкретної моделі, включно з URL, розміром, мовою та іншими параметрами. Наприклад:

https://awsde0.fds.api.xiaomi.com/xiaomi-product/xiaomi.vacuum.b108gl/voices/soundpackage.json
https://awsde0.fds.api.xiaomi.com/dreame-product/xiaomi.vacuum.b108gl/voices/soundpackage.json

При розробці серверної частини платформи доцільно реалізувати регулярний сканер таких ресурсів для актуалізації метаданих про офіційні голосові пакети. Це дозволяє автоматично додавати нові варіанти озвучок до каталогу, не очікуючи ручного втручання.

Особливу увагу варто приділити питанню формування зв’язку між моделлю пилососа та відповідним форматом озвучки. У базі необхідно підтримувати відповідність між ідентифікатором моделі (наприклад dreame.vacuum.p2029) і типом пакування, який підтримує ця модель (наприклад ogg/tar.gz). Таким чином,

при спробі завантажити голосовий пакет користувачу не пропонується випадковий файл, а саме той, який технічно сумісний із його пристроєм.

Для оптимізації запитів з боку вебінтерфейсу платформи доцільно зберігати всі ці метадані у форматі, зручному для API — наприклад, у вигляді REST-запиту, який повертає JSON-об’єкт з переліком доступних озвучок, фільтрацією за мовою, стилем, моделлю та форматом. Це дозволяє фронтенду швидко сформувати інтерфейс для завантаження або кастомізації озвучок. [47]

Таким чином, каталогізація та зберігання озвучок — це не лише зручність користувача, а й технічна основа для стабільної взаємодії з розумними пристроями, яка забезпечує коректне та безпечне оновлення їхнього голосового інтерфейсу. Це також створює підґрунтя для майбутнього розширення платформи на інші типи пристроїв, зберігаючи при цьому єдиний підхід до структури та логіки зберігання голосових даних.

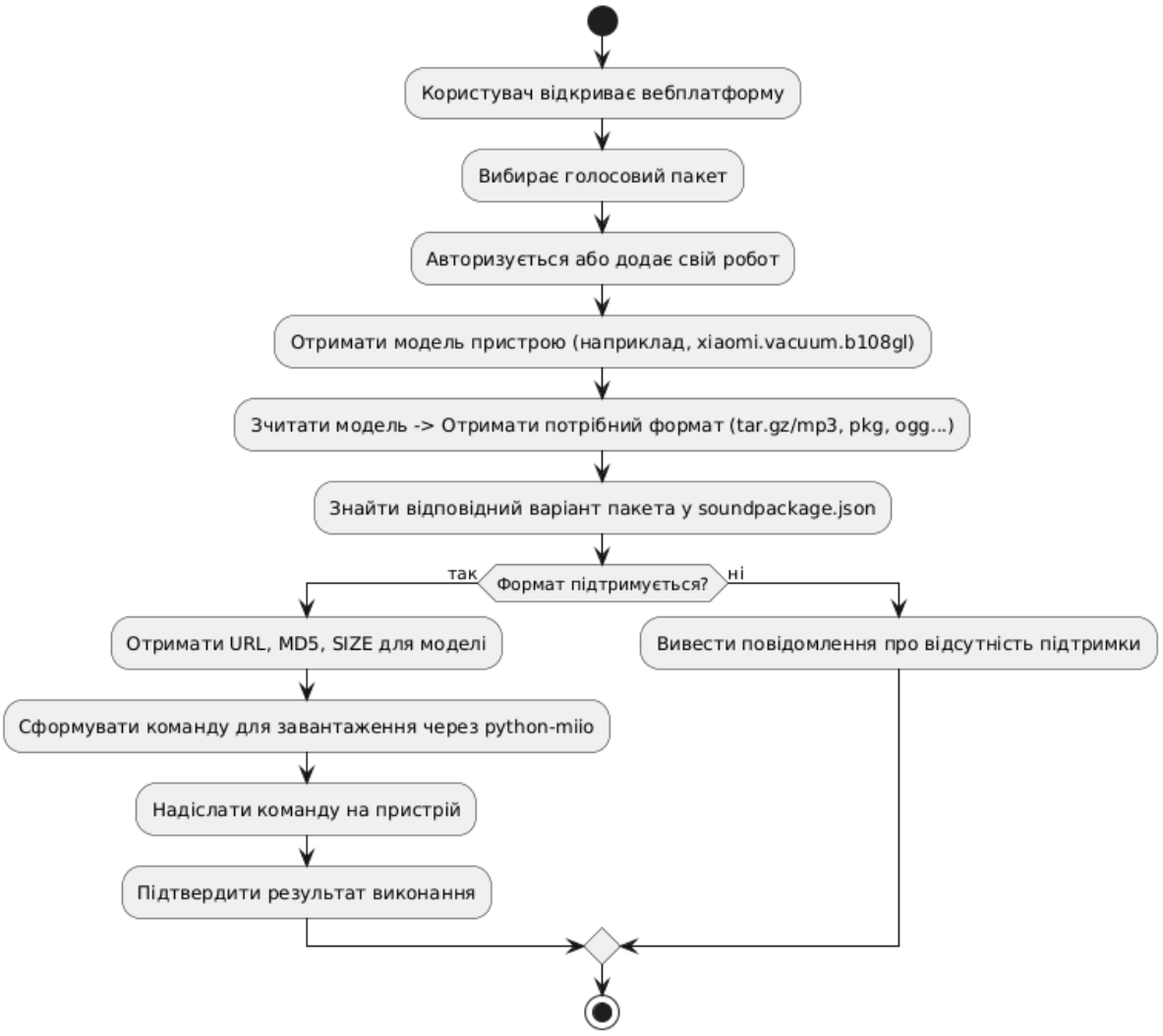


Рисунок 3.10 Діаграма взаємодії користувача з платформою завантаження голосових пакетів для роботів-пилососів

3.8 Розробка платформи створення та завантаження користувацьких голосових пакетів для роботів-пилососів

Одним із принципово важливих викликів при реалізації платформи кастомізації голосових повідомлень для роботів-пилососів є підтримка відмінностей у форматах голосових пакетів, що використовуються різними виробниками. На відміну від уніфікованих підходів до озвучення, кожен бренд реалізує зберігання та передачу голосових файлів за власною логікою. Наприклад, для пристроїв Roborock типовим є формат .pkg, що складається з набору бінарних аудіофайлів у структурованому архіві, який має бути завантажено через спеціалізовану команду. У випадку з більшістю моделей Xiaomi, які не належать до Roborock або Dreame, використовується формат .tar.gz, у якому містяться .mp3 файли з чіткою прив’язкою до системних подій. Для пристроїв Dreame також застосовується формат .tar.gz, але всередині архіву використовуються файли у форматі .ogg. Ще складнішим є сценарій для окремих моделей, таких як ijai.vacuum.v3, що поширені на материковому ринку Китаю — у них використовується власний тип пакету, специфікація якого часто не є відкритою або стандартизованою.

У зв’язку з цим платформа повинна мати механізм адаптації єдиного логічного голосового пакету до різних форматів залежно від моделі пристрою користувача. З технічної точки зору, це означає, що при збереженні голосового пакету до системи створюється логічний шаблон, а для кожного підтримуваного типу пристрою формується відповідна збірка у правильному форматі. Це дозволяє уникнути дублювання контенту на рівні інтерфейсу та зробити користувацький досвід більш цілісним. Коли користувач завантажує певний голосовий пакет, система автоматично визначає модель пристрою і надає правильну варіацію пакету: .pkg, .tar.gz з .mp3, .tar.gz з .ogg, або інший, специфічний для моделі формат.

Процес завантаження голосових пакетів на пристрій також може відрізнятись залежно від архітектури пристрою та способу реалізації API. Згідно з документацією платформи MIOT, деякі пристрої використовують properties з атрибутом ppid, інші — actions з атрибутом aaid. Таким чином, платформа має включати логіку аналізу MIOT-специфікації на основі моделі пристрою. Це може бути реалізовано у вигляді окремого модуля, який парсить опис функцій з сайту https://home.miot-spec.com, зберігає відповідну схему у базу даних, а під час завантаження голосового пакету обирає правильний метод на основі цієї інформації.

З метою максимальної автоматизації платформа також повинна підтримувати авторизацію в екосистемах Xiaomi Home або Dreame Home для зчитування списку пристроїв, доступних у обліковому записі користувача. Для цього використовується інструмент miiocli, який дозволяє отримати перелік пристроїв разом із моделлю, версією, токеном доступу та іншими метаданими через команду miiocli device list. У межах веб-інтерфейсу необхідно реалізувати форму авторизації, де користувач вводить логін і пароль до свого облікового запису. У безпечному середовищі сервер виконує вхід, отримує JSON-відповідь від Xiaomi Cloud та парсить список пристроїв у форматі xiaomi.vacuum.b108gl, dreame.vacuum.r2394s, roborock.vacuum.a10 тощо.

Визначення актуальної IP-адреси пристрою виконується через локальний сканер мережі або API-запити до шлюзу. На основі токена та IP-адреси виконується зв’язок із пристроєм, і платформа здатна не лише завантажити голосовий пакет, але й зчитати наявні параметри, наприклад, версію прошивки, поточний стан пристрою чи активний голосовий пакет.

Для забезпечення контролю цілісності кожен голосовий пакет повинен супроводжуватись метаданими, де зберігається його унікальна контрольна сума (MD5), URL для завантаження та точний розмір у байтах. Ці метадані зберігаються в базі даних і використовуються як для верифікації пакета перед завантаженням, так і для порівняння при оновленнях.

Додатковим технічним викликом є спосіб запуску команд python-miio. У початковій версії планується використовувати локальний запуск через API-обгортку у бекенді. Проте для подальшої масштабованості та автоматизації оновлень доцільно передбачити варіант запуску цього скрипта через GitHub Actions. Завдяки цьому підхід дозволяє виконувати генерацію або завантаження голосових пакетів як CI/CD-процеси — наприклад, кожного разу, коли створюється новий логічний шаблон голосового пакету, GitHub Actions автоматично будує .pkg, .tar.gz і .ogg варіанти, записує їх у файлове сховище, оновлює базу метаданих і сповіщає користувача про готовність завантаження. Таке рішення не лише пришвидшує цикл генерації контенту, але й розвантажує сервер у пікові моменти.

Платформа, що об’єднує в собі динамічну генерацію, адаптивне форматування, багатоканальне завантаження та інтеграцію з хмарними сервісами екосистеми Xiaomi, створює нову якість у сфері персоналізації побутових пристроїв. У перспективі вона може стати основою для подібних рішень у інших категоріях техніки — не лише для голосових повідомлень, а й для прошивок, конфігурацій та звукових сигналів на основі поведінкових сценаріїв.

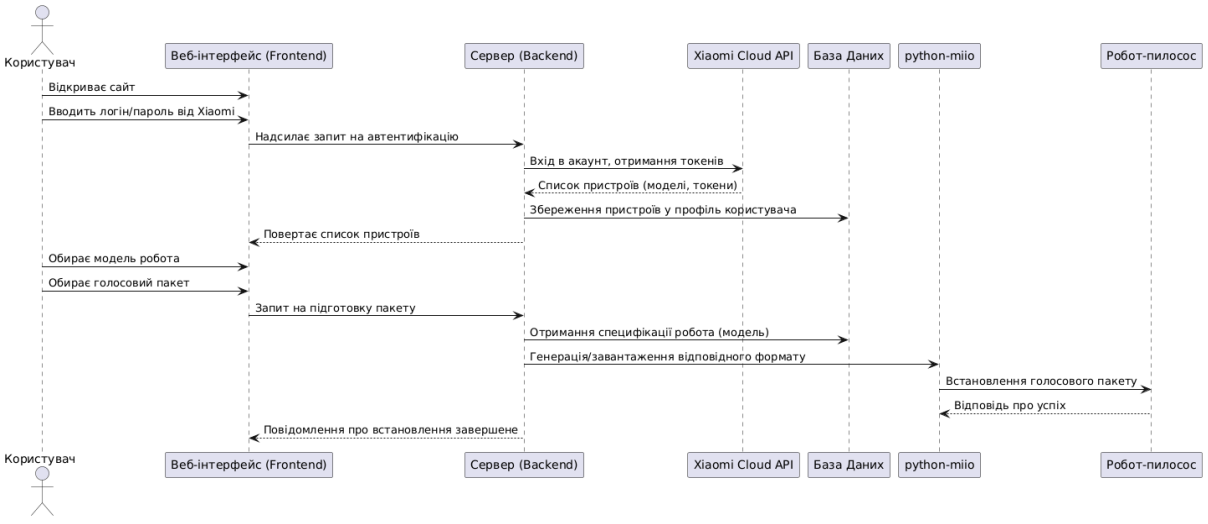


Рисунок 3.11 Діаграма послідовності (авторизація, вибір робота, генерація та завантаження пакету) [48]

Додатковий функціонал платформи полягає у генерації та пакуванні голосових файлів за допомогою GitHub Actions. Це дозволяє виконувати процес створення голосових пакетів автоматично, без необхідності встановлювати

додаткове програмне забезпечення на власному комп’ютері. Основа цього функціоналу — інструмент audio_generator із відкритого проєкту Dustcloud, який підтримує генерацію озвучених фраз на основі CSV-файлів. У таких CSV-файлах зберігаються текстові фрази, які відповідають командам робота-пилососа, наприклад «починаю прибирання», «зарядка» або «прибирання завершено». Для кожної мови створюється окремий CSV-файл зі стандартним набором фраз. Користувач може переглянути цей файл, а за бажанням — змінити його, додавши власні унікальні фрази. Після цього платформа запускає процес генерації, у результаті якого створюються готові голосові пакети у потрібному форматі — .pkg, .tar.gz з MP3 або OGG файлами — відповідно до типу робота-пилососа. Таким чином, платформа надає не лише зручний інтерфейс для вибору або завантаження пакетів, а й повноцінну систему для їхнього створення, яку можна використовувати як з готовими шаблонами, так і з власними текстами.

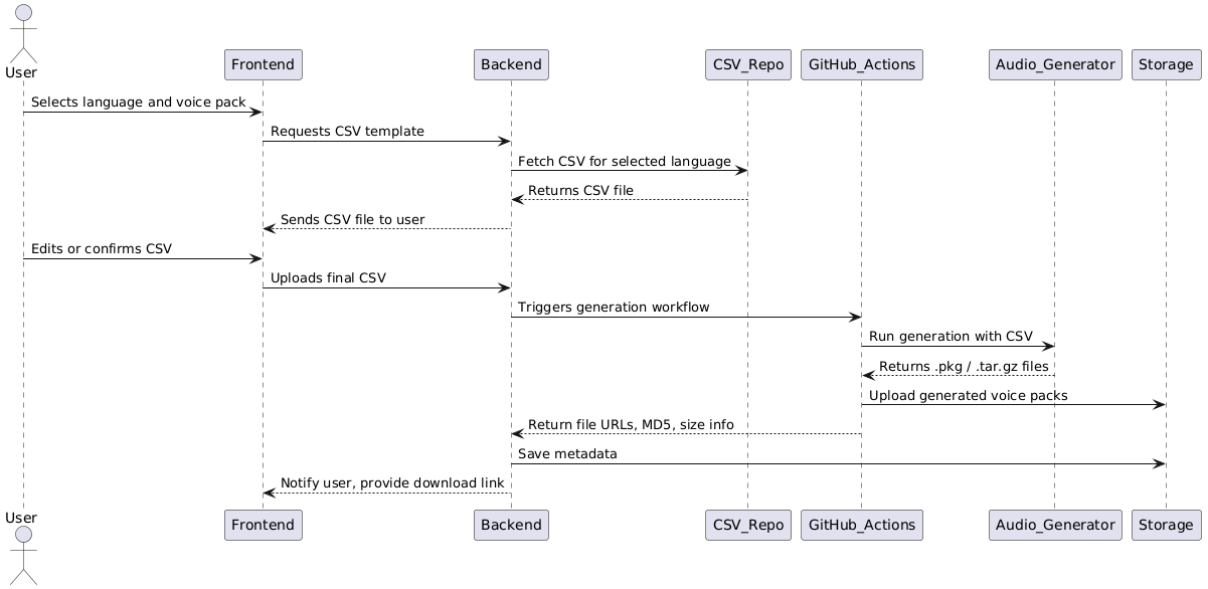


Рисунок 3.12 Взаємодія користувача з Frontend та Backend частиною платформи

ВИСНОВОК

У ході виконання цієї дипломної роботи було досліджено та практично реалізовано підходи до модифікації функціональності роботів-пилососів за допомогою відкритих програмних інструментів. Сучасна побутова техніка, зокрема роботи-пилососи, перестали бути лише допоміжними пристроями — вони дедалі більше інтегруються в загальну екосистему розумного дому, стаючи повноцінними учасниками автоматизованих сценаріїв. У цій роботі я не просто розглянув наявні рішення, а спробував на практиці реалізувати платформу, яка дозволяє розширювати можливості роботів без прив’язки до закритих серверів виробників. Це відкриває нові горизонти для користувача — як з погляду кастомізації, так і з позиції конфіденційності, контролю та свободи.

Починаючи з вивчення архітектури типових моделей роботів-пилососів, я зосередив увагу на прошивках, які надають доступ до управління пристроєм через локальні засоби — зокрема таких рішень, як Valetudo та Congatudo. Їхній відкритий код, активна спільнота розробників та можливість гнучкої інтеграції через MQTT дозволили створити систему, яка не лише повторює базовий функціонал виробника, але й значно розширює його. Зокрема, можливість прямої взаємодії з Home Assistant дала змогу автоматизувати поведінку робота в залежності від подій у домі, часу доби чи присутності людей.

Особливу увагу було приділено розробці кастомізованих голосових пакетів. Робот-пилосос, який озвучує свою роботу індивідуально підібраними фразами, стає не просто технікою, а елементом персонального простору. Було створено базовий інструментарій для генерації та завантаження таких пакетів, що, в перспективі, може бути використано ширше — не лише для зміни голосу, а й для перекладу інтерфейсу, створення мультиголосових сценаріїв або імітації мовлення конкретних персонажів. Окрім розважального ефекту, це також слугує потужним інструментом інклюзії для користувачів із особливими потребами.

У ході реалізації я також ознайомився з такими фундаментальними технологіями, як ROS, OpenCV, Webots. Хоча більшість комерційних пилососів не

працюють безпосередньо з цими платформами, саме вони формують основу відкритої робототехніки, і розуміння їх принципів допомогло глибше осмислити можливості майбутньої інтеграції більш складних алгоритмів, зокрема візуального позиціонування, виявлення об’єктів, адаптивної навігації. Деякі ідеї, які не вдалося реалізувати в рамках цього дослідження через обмеження апаратного забезпечення, вже заклали напрямок для майбутніх розробок — наприклад, використання зовнішніх процесорних модулів на кшталт Raspberry Pi як розширень до комерційних роботів.

Важливою частиною роботи було також документування процесів — починаючи з аналізу прошивки, структури MQTT-повідомлень, принципів синтезу мовлення — і завершуючи оформленням користувацької інструкції для створеної платформи. Це дозволяє іншим ентузіастам повторити реалізовані рішення, адаптувати їх під власні потреби або, навпаки, почати свій шлях у світі відкритої робототехніки. Зрештою, саме відкритість та спільна робота — основа технологічного поступу в цьому напрямку.

Підсумовуючи, можу сказати, що ця дипломна робота стала не лише академічним проєктом, а й особистим викликом: чи можна зробити щось більше, ніж просто користуватись готовим пристроєм? Чи можна повернути собі контроль над технікою, зробити її не тільки корисною, але й «своєю»? У процесі дослідження я переконався, що це цілком можливо — і навіть більше: доступні технології вже дозволяють створювати нові функції, експериментувати зі взаємодією людини і машини, прокладати місток між побутовою технікою і творчістю.

Відкрита робототехніка — це не лише про код, залізо та протоколи. Це про культуру відкритості, експерименту і співпраці. І я радий, що зробив свій внесок у цю справу, хай і на рівні бакалаврського дослідження. Робота з такими інструментами — це завжди трохи більше, ніж просто навчання: це спосіб мислити, створювати й уявляти майбутнє, де технології справді працюють на людину — і в найпрямішому сенсі слухають її голос.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] The best robot vacuum for me is the one I hacked | The Verge https://www.theverge.com/23934731/valetudo-robot-vacuum-hacking

[2] How to install Valetudo RE on a Xiaomi robot vacuum the easy way. https://www.reddit.com/r/homeassistant/comments/fdrcz0/how_to_install_valetudo_re_on_a_xiaomi_robot/

[3] ``Valetudo'' allows you to remove the cloud connection from your ... https://gigazine.net/gsc_news/en/20231229-valetudo/

[4] I've made a RoboRock (Valetudo-based vacuum robot) voice lines ... https://community.home-assistant.io/t/ive-made-a-roborock-valetudo-based-vacuum-robot-voice-lines-generator/626429

[5] Blueprints for using Valetudo with Home Assistant - GitHub https://github.com/mundschenk-at/ha-valetudo-blueprints

[6] Valetudo Dreame Z10 Pro Voice Packs Guide - GitHub Gist https://gist.github.com/xmesaj2/33af535ee9eb37c947135042c4ebb4c0

[7] Capabilities Overview - Valetudo https://valetudo.cloud/pages/usage/capabilities-overview.html

[8] MQTT - Valetudo https://valetudo.cloud/pages/integrations/mqtt.html

[9] Home Assistant - Valetudo https://valetudo.cloud/pages/integrations/home-assistant-integration.html

[10] How to integrate xiaomi vaccum with Valetudo to Home Assistant? https://community.home-assistant.io/t/how-to-integrate-xiaomi-vaccum-with-valetudo-to-home-assistant/114858

[11] Hypfer/Valetudo: Cloud replacement for vacuum robots ... - GitHub https://github.com/Hypfer/Valetudo

[12] Those running a robot with Valetudo, what's your experience like ... https://www.reddit.com/r/RobotVacuums/comments/18x6fwy/those_running_a_robot_with_valetudo_whats_your/

[13] Building and Modifying Valetudo https://valetudo.cloud/pages/development/building-and-modifying-valetudo.html

[14] De-Clouding a Robot Vacuum with Valetudo! Roborock Q7 Max+ https://www.youtube.com/watch?v=qHuCcxR9jNc&lc=UgyzK_MYUYCf18BaZJF4AaABAg

[15] Valetudo: Useful blueprints for your vacuum robots https://community.home-assistant.io/t/valetudo-useful-blueprints-for-your-vacuum-robots/437063?page=2

[16] Valetudo: Useful blueprints for your vacuum robots https://community.home-assistant.io/t/valetudo-useful-blueprints-for-your-vacuum-robots/437063

[17] FAQ - Valetudo https://valetudo.cloud/pages/faq.html

[18] Roborock S50 with Valetudo RE firmware and MQTT integration https://community.openhab.org/t/roborock-s50-with-valetudo-re-firmware-and-mqtt-integration/101317

[19] Valetudo Companion - Apps on Google Play https://play.google.com/store/apps/details?id=cloud.valetudo.companion

[20] Valetudo: Taming spying automatic robot vacuum cleaner - BornCity https://borncity.com/win/2024/11/03/valetudo-taming-spying-automatic-robot-vacuum-cleaner/

[21] Valetudo 2024.06.2 · Hypfer Valetudo · Discussion #2078 - GitHub https://github.com/Hypfer/Valetudo/discussions/2078

[22] Valetudo – Cloud replacement for vacuum robots enabling local ... https://news.ycombinator.com/item?id=38788326

[23] valetudo-sunxi-livesuit/README.md at master - GitHub https://github.com/Hypfer/valetudo-sunxi-livesuit/blob/master/README.md

[24] Getting Started - Valetudo https://valetudo.cloud/pages/general/getting-started.html

[25] Voice packs not uploading via Valetudo GUI - Lemmy.World https://lemmy.world/post/18567576

[26] Valetudo Map Card Installation Instructions - Third party integrations https://community.home-assistant.io/t/valetudo-map-card-installation-instructions/247329

[27] De-Clouding a Robot Vacuum with Valetudo! Roborock Q7 Max+ https://www.youtube.com/watch?v=qHuCcxR9jNc

[28] Rooted Roborock S7 running Valetudo - Voice Packs - Reddit https://www.reddit.com/r/Roborock/comments/tfl9wd/rooted_roborock_s7_running_valetudo_voice_packs/

[29] Valetudo Map Card Installation Instructions - #11 by PsyGanja https://community.home-assistant.io/t/valetudo-map-card-installation-instructions/247329/11

[30] Mi Robot Vacuum. Valetudo RE 0.9.8. Установка, обзор и настройка. https://www.youtube.com/watch?v=AFlqR-4rkOw

[31] GLaDOS (Portal) voice pack for Roborock / Xiaomi vacuums https://community.home-assistant.io/t/glados-portal-voice-pack-for-roborock-xiaomi-vacuums/241101

[32] Valetudo Newcomer Guide https://valetudo.cloud/pages/general/newcomer-guide.html

[33] Hypfer/valetudo-crl200s-root: Tooling to root CRL-200S ... - GitHub https://github.com/Hypfer/valetudo-crl200s-root

[34] Supported Robots - Valetudo https://valetudo.cloud/pages/general/supported-robots.html

[35] Valetudo | Cloud replacement for vacuum robots enabling local-only ... https://valetudo.cloud

[36] Valetudo: Cloud replacement for vacuum robots enabling local-only ... https://www.reddit.com/r/selfhosted/comments/1iwe02u/valetudo_cloud_replacement_for_vacuum_robots/

[37] Roborock Vacuum with Home Assistant - YouTube https://www.youtube.com/watch?v=WX2bP_Nx3Uc

[38] Home Assistant - Congatudo https://congatudo.cloud/pages/integrations/home-assistant-integration.html

[39] Segment/room cleaning using Valetudo (and Roborock S5) #732 https://github.com/PiotrMachowski/lovelace-xiaomi-vacuum-map-card/discussions/732

[40] Valetudo Automations : r/homeassistant - Reddit https://www.reddit.com/r/homeassistant/comments/1bu4qq6/valetudo_automations/

[41] Valetudo RE: (Almost) fully working example #435 - GitHub https://github.com/PiotrMachowski/lovelace-xiaomi-vacuum-map-card/discussions/435

[42] Valetudo is a cloud-free web interface for robot vacuum cleaners https://www.cnx-software.com/2021/01/20/valetudo-is-a-cloud-free-web-interface-for-robot-vacuum-cleaners/

[43] README.md - Hypfer/valetudo-crl200s-root - GitHub https://github.com/Hypfer/valetudo-crl200s-root/blob/master/README.md

[44] Rooting and Installing Valetudo on Roborock Vacuum Cleaners https://daanmiddendorp.com/tech/2023/01/08/rooting-a-vacuum-cleaner

[45] Howto https://builder.dontvacuum.me/howtos/1c/installer/_howto.html

[46] Free your robot vacuum from the cloud - ajfriesen https://www.ajfriesen.com/cloud-free-robot-vacuum/

[47] Custom Voice · rand256 valetudo · Discussion #361 - GitHub https://github.com/rand256/valetudo/discussions/361

[48] Valetudo 2024.06.0 · Hypfer Valetudo · Discussion #2073 - GitHub https://github.com/Hypfer/Valetudo/discussions/2073