ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«МОДЕЛЬ СПЕЦІАЛІЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ОБ'ЄКТАМИ ІОТ МЕРЕЖ»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми «Інформаційні системи та технології»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Олександр ТЕРЕХОВ*
(підпис) (Ім'я, ПРІЗВИЩЕ здобувача)

Виконав: здобувач вищої освіти гр. ІСД-41
Олександр ТЕРЕХОВ
(Ім'я, ПРІЗВИЩЕ)

Керівник: Андрій БОНДАРЧУК
науковий ступінь, вчене звання (Ім'я, ПРІЗВИЩЕ)

Рецензент: _____
науковий ступінь, вчене звання (Ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти бакалавр

Спеціальність 126 Інформаційні системи та технології

Освітньо-професійна програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСТ
_____ Каміла СТОРЧАК
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Терехов Олександр Сергійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Модель спеціалізованої системи управління об'єктами IoT мереж»
керівник кваліфікаційної роботи: Андрій БОНДАРЧУК, доктор технічних наук, професор ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. №56
2. Строк подання кваліфікаційної роботи «30» травня 2025р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, документація платформи Home Assistant, технічні специфікації пристроїв Tuya Smart, стандарти локальних протоколів передачі даних (HTTP, MQTT, Local API), методи локальної інтеграції IoT-пристроїв, структура автоматизації в системах розумного дому, вимоги до побудови автономної мережі управління, приклади сценаріїв автоматизації та принципи організації безпечного локального середовища.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Теоретичні основи побудови моделі спеціалізованої IoT-системи
    2. Архітектура та реалізація моделі спеціалізованої системи управління
    3. Оптимізація, оцінка та напрями розвитку системи
5. Перелік ілюстративного матеріалу: *презентація*
    1. Мета, об'єкт, предмет дослідження
    2. Актуальність роботи
    3. Попередня архітектура на базі Tuya Smart Life
    4. Проблематика існуючої реалізації
    5. Архітектура запропонованого рішення: Home Assistant та інтеграція LocalTuya
    6. Розгортання системи на базі Home Assistant OS (VirtualBox)
    7. Інтеграція існуючих пристроїв: Tuya та LocalTuya
    8. Згенерований дашборд користувача після хмарної інтеграції
    9. Кастомізований дашборд користувача після локальної інтеграції
    10. Алгоритми комбінованих сценаріїв автоматизації

11. Аналіз автономної роботи реалізованої системи
12. Порівняння продуктивності двох систем
13. Кошторис реалізації впровадженої системи
14. Висновки
15. Публікації та апробація роботи

6. Дата видачі завдання «24» лютого 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	25.02-19.03.2025	
2	Обробка матеріалу	20.03-02.04.2025	
3	Розробка теоретичної частини	03.04-07.04.2025	
5	Архітектура та реалізація моделі спеціалізованої системи управління	08.04-27.04.2025	
6	Оптимізація, оцінка та напрями розвитку системи	28.04-11.05.2025	
7	Висновки за результатами аналізу	12.05-15.05.2025	
8	Розробка презентації	16.05-26.05.2025	
9	Передзахист	27.05-29.05.2025	
10	Здача в деканат	30.05-05.06.2025	

Здобувач вищої освіти _____ Олександр ТЕРЕХОВ
(підпис) (Ім’я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ Андрій БОНДАРЧУК
(підпис) (Ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 93 стор., 46 рис., 7 табл., 37 джерел.

Мета роботи – спрощення локального управління пристроями IoT-мереж за рахунок використання принципів автономної інтеграції та автоматизації.

Об'єкт дослідження – процес локальної взаємодії та керування побутовими IoT-пристроями в інфраструктурі розумного будинку.

Предмет дослідження – засоби локальної інтеграції та автоматизації з використанням відкритих протоколів і платформ.

Короткий зміст роботи: У роботі розглянуто можливості платформи Home Assistant для автономного керування Wi-Fi-пристроями екосистеми Tuya. Проаналізовано два підходи інтеграції: хмарний (Tuya IoT Cloud та Smart Life) та локальний (Home Assistant та LocalTuya), із порівнянням їхньої ефективності, затримки реакції та залежності від інтернет-з'єднання.

Реалізовано комбіновані сценарії автоматизації керування освітленням та енергоспоживанням, що дозволило оцінити стабільність та зручність локального управління. Визначено переваги локального підходу, зокрема автономність, швидкодію та безпеку при мінімальній складності впровадження.

КЛЮЧОВІ СЛОВА: РОЗУМНИЙ ДІМ, IOT, HOME ASSISTANT, TUYA, LOCALTUYA, АВТОМАТИЗАЦІЯ, ЛОКАЛЬНЕ КЕРУВАННЯ, ІНТЕГРАЦІЯ, СЦЕНАРІЇ.

ABSTRACT

Text part of the qualification work for obtaining a bachelor's degree: 93 pages, 46 figures, 7 tables, 37 sources.

The purpose of the work is to simplify local management of IoT network devices by using the principles of autonomous integration and automation.

The object of the study is the process of local interaction and management of household IoT devices in the smart home infrastructure.

The subject of the study is local integration and automation tools using open protocols and platforms.

Summary of the work: The paper examines the capabilities of the Home Assistant platform for autonomous management of Wi-Fi devices in the Tuya ecosystem. Two integration approaches are analyzed: cloud (Tuya IoT Cloud and Smart Life) and local (Home Assistant and LocalTuya), with a comparison of their efficiency, response delay and dependence on the Internet connection.

Combined scenarios for automation of lighting and energy consumption control are implemented, which made it possible to assess the stability and convenience of local management. The advantages of the local approach are identified, in particular autonomy, speed and security with minimal complexity of implementation.

KEYWORDS: SMART HOME, IOT, HOME ASSISTANT, TUYA, LOCALTUYA, AUTOMATION, LOCAL CONTROL, INTEGRATION, SCENARIOS.

ЗМІСТ

ВСТУП 9
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ МОДЕЛІ СПЕЦІАЛІЗОВАНОЇ ІОТ-СИСТЕМИ 10
1.1 Інформаційні технології як основа автоматизації середовищ з об’єктами IoT 10
1.2 Функціональна модель системи управління об’єктами в IoT-середовищі 15
1.3 Принципи побудови інформаційної взаємодії у хмарному середовищі 21
1.4 Архітектура існуючої системи управління побутовими пристроями 29
1.5 Причини переходу до відкритої системи 38
РОЗДІЛ 2. АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ МОДЕЛІ СПЕЦІАЛІЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ 43
2.1 Аналіз керованих об’єктів і умов їх експлуатації 43
2.2 Архітектура кастомізованої системи керування 49
2.3 Реалізація логіки управління та автоматизацій 66
2.4 Тестування системи в умовах автономної роботи 74
РОЗДІЛ 3. ОПТИМІЗАЦІЯ, ОЦІНКА ТА НАПРЯМИ РОЗВИТКУ СИСТЕМИ ... 81
3.1 Порівняльна оцінка реалізованої системи з хмарною моделлю 81
3.2 Аналіз альтернатив і оцінка вибору платформи 88
3.3 Перспективи вдосконалення побудованої системи 95
ВИСНОВКИ 101
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 102
Додаток А. Конфігурація дашборда в Home Assistant для керування побутовими пристроями 105
Додаток Б. Комбінований сценарій автоматизації у Home Assistant: автокерування при вході/виході з дому з перевіркою часу 108
Додаток В. Комбінований сценарій автоматизації у Home Assistant: автоматизація світла ввечері та вночі 109
Додаток Г. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) 110

ВСТУП

Інтернет речей поступово трансформує повсякденне життя, роблячи керування освітленням, безпекою, кліматом та іншими побутовими процесами зручнішим і гнучкішим. Смарт-пристрої, з’єднані в єдину систему, створюють базу для побудови автоматизованих сценаріїв, які покращують комфорт і ефективність. Утім, зростає попит на рішення, що забезпечують не лише функціональність, а й приватність, автономність та можливість локального контролю, без залежності від сторонніх хмарних сервісів.

Типовим прикладом початкової реалізації IoT-системи є використання Wi-Fi-пристроїв, інтегрованих у платформу Smart Life від екосистеми Tuya. Такий підхід дозволяє швидко налаштувати керування через мобільний додаток і створити прості автоматизації. Водночас хмарна модель має суттєві обмеження щодо складності логіки, швидкості реакції, стабільності при втраті з’єднання та захисту даних, що знижує її доцільність у більш складних або критичних застосуваннях.

Альтернативою стає використання Home Assistant — універсальної локальної платформи з відкритим кодом, яка підтримує як хмарні, так і локальні інтеграції. Можливість створення складних сценаріїв через YAML-конфігурації, обробка історичних даних, взаємодія з зовнішніми API та розширена логіка автоматизації робить її гнучким рішенням для різних задач. Крім того, платформа дозволяє налаштовувати зберігання даних, резервне копіювання, доступ через локальну мережу та контроль стабільності живлення.

З огляду на зростаючі вимоги до автономності, гнучкості та безпеки, перехід на платформу Home Assistant виглядає виправданим, що дозволить побудувати стабільну систему управління IoT, здатну працювати без доступу до інтернету, зберігаючи повну функціональність і контроль над усіма компонентами, що є ключовим для реалізації сучасної домашньої або комерційної IoT-інфраструктури.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ МОДЕЛІ СПЕЦІАЛІЗОВАНОЇ ІОТ-СИСТЕМИ

1.1 Інформаційні технології як основа автоматизації середовищ з об’єктами IoT

Інтернет речей у своєму сучасному розумінні є концепцією цифрової трансформації, яка активно розвивається і передбачає інтеграцію фізичних об’єктів у єдину мережу з можливістю збирання, обміну та обробки даних, при цьому самі об’єкти зазвичай оснащені сенсорами, контролерами та інтерфейсами безпровідного зв’язку, що дає змогу їм взаємодіяти з іншими системами без безпосередньої участі людини, забезпечуючи гнучке й ефективне управління ресурсами, моніторингом і автоматизацією у багатьох сферах діяльності. Така архітектура дозволяє реалізовувати динамічні рішення у реальному часі, що особливо важливо для сценаріїв, де критичною є швидкість реакції на зміну стану середовища, споживання ресурсів або поведінки користувачів [1-4].

Основні напрями застосування IoT у житлових і виробничих просторах. Інтернет речей широко впроваджується у житлових і виробничих просторах, зокрема в концепції «розумного дому», де використовується для створення систем:

— комфортного керування освітленням;
— енергоспоживанням;
— безпекою;
— мікрокліматом.

Наприклад, розумні лампи дозволяють змінювати яскравість і колір освітлення дистанційно або за розкладом, що дає змогу адаптувати умови освітлення під потреби користувача в різний час доби. Розумні розетки, оснащені функціями моніторингу, допомагають виявляти неефективне використання електроенергії, дозволяючи вмикати та вимикати побутові прилади дистанційно або

автоматизовано, оптимізуючи витрати на електроенергію. Датчики витоку води або газу автоматично фіксують аварійні ситуації та надсилають повідомлення, а також можуть активувати заздалегідь запрограмовані сценарії захисту, що суттєво знижує ризики пошкоджень майна чи загроз для життя мешканців [1].

У виробничому середовищі інтернет речей насамперед орієнтований на:

— автоматизацію технологічних процесів;

— дистанційний моніторинг стану обладнання;

— прогнозування технічного обслуговування;

— аналітику виробничих показників.

Такі системи дають змогу створювати інтелектуальні рішення для відстеження температури, вологості, рівня газів та інших фізичних параметрів, що впливають на ефективність та безпеку виробничих процесів. Завдяки безперервному збору даних з датчиків стає можливим своєчасне виявлення зношування устаткування, оптимізація графіків обслуговування, зменшення простоїв і підвищення загальної продуктивності, водночас знижуючи потребу в ручному контролі з боку персоналу. У такий спосіб впровадження IoT-технологій у виробничій сфері трансформує звичайні підприємства на основі даних, дозволяючи швидше ухвалювати рішення, підвищувати точність і ефективність роботи [1].

У даному підтексті, інтернет речей розглядається не просто як окрема технологія, а як складова загальної стратегії цифровізації середовища, яка охоплює як побутові, так і промислові простори. Його масштабованість, гнучкість і відносна доступність відкривають широкі можливості для адаптації в різних умовах, де критичним є баланс між зручністю користування, автоматизацією процесів і економічною ефективністю [1].

Переваги автоматизованого управління побутовими пристроями. Автоматизація управління побутовими пристроями в умовах житлового середовища стала можливою завдяки активному поширенню доступних мікроконтролерів, сенсорів і безпровідних комунікацій, а також завдяки мобільним застосункам, які забезпечують інтуїтивний інтерфейс керування. У межах побутових рішень така автоматизація виконує цілу низку функцій, спрямованих на:

— підвищення якості життя;

— підсилення безпеки;

— раціональне використання ресурсів.

Однією з найважливіших переваг є зручність, адже пристрої працюють згідно з заданими сценаріями без необхідності постійного втручання користувача — наприклад, розетки можуть вмикати побутову техніку в певний час або вимикати її при досягненні ліміту енергоспоживання, що особливо корисно для усунення прихованих витрат у режимі очікування [2].

Разом з цим автоматизація покращує загальну безпеку, оскільки, наприклад, детектори витоку води чи газу здатні оперативно повідомити про загрозу через сповіщення або навіть автоматично запустити аварійні механізми на кшталт перекриття подачі води або газу, якщо в систему включено відповідне обладнання. Завдяки цьому зменшується ймовірність матеріальних збитків, а також ризик для життя мешканців. Ще одним суттєвим аспектом є енергоефективність, адже автоматизоване керування освітленням і побутовими приладами дозволяє оптимізувати споживання електроенергії через використання гнучких розкладів, сценаріїв вимкнення або зонального освітлення, що в довгостроковій перспективі сприяє зменшенню витрат на комунальні послуги. Інтеграція з мобільними застосунками додатково відкриває можливість постійного дистанційного контролю, незалежно від місця перебування користувача, що загалом підвищує рівень комфорту й контролю в побуті без потреби в складних технічних навичках чи значних фінансових витратах [2].

Роль комерційних екосистем у спрощенні впровадження. Важливою складовою, що суттєво вплинула на поширення подібних рішень, стали комерційні екосистеми, зокрема Tuya (див. рис. 1.1), яка забезпечує повноцінну інфраструктуру для виробників і кінцевих користувачів. Такі платформи включають:

— хмарні сервіси;

— мобільні додатки;

— бібліотеки SDK;

— засоби автоматизації.

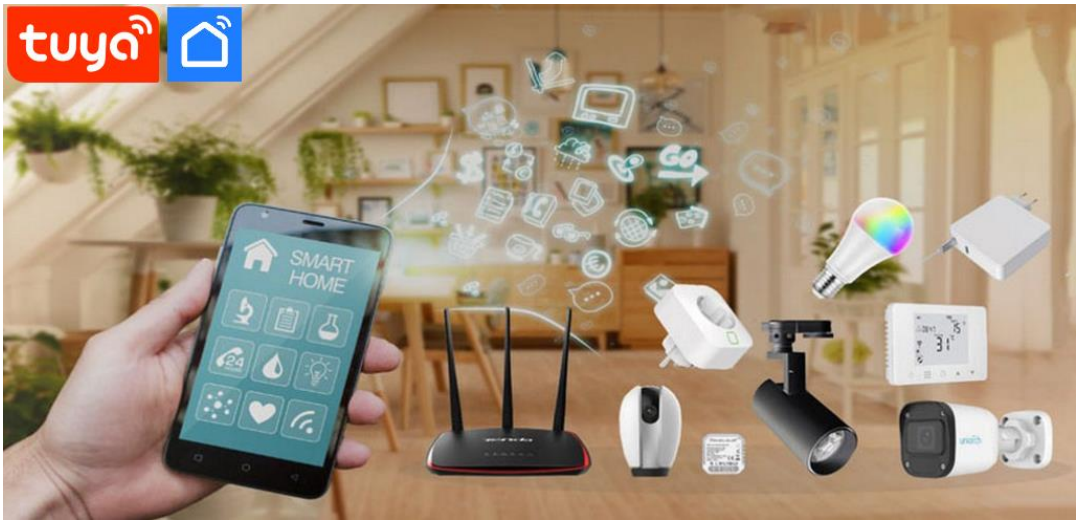

Рисунок 1.1 – Екосистема керування IoT пристроями: Tuya

У контексті Tuya реалізовано стандартизований підхід до інтеграції та керування різноманітними пристроями, що підтримують сумісність з цією платформою, завдяки чому можливе поєднання техніки різних брендів в одному інтерфейсі – це так званий підхід "White Label", за якого виробники лише адаптують свої пристрої під загальну хмарну інфраструктуру, зберігаючи при цьому сумісність з основними функціями.

Для користувача такий підхід означає:

— легке додавання нових пристроїв через мобільний застосунок;

— доступ до готових шаблонів сценаріїв;

— можливість керування за допомогою голосових помічників;

— використання візуального редактора сценаріїв.

Додатково для технічних ентузіастів і розробників платформа Tuya пропонує розробницьке середовище з доступом до аналітики, логіки роботи пристроїв і створення власних кастомних інтеграцій через Tuya IoT Platform. Такий комплексний підхід дозволяє суттєво скоротити час і зусилля, необхідні для створення повноцінної автоматизованої системи управління побутовими

пристроями, і робить реалізацію концепції розумного будинку доступною навіть на базовому рівні для широкого кола користувачів [3].

Недоліки закритих платформ і потреба у відкритих рішеннях. Попри численні переваги використання закритих комерційних платформ, таких як Tuya, на практиці виникають суттєві обмеження, які особливо помітні при масштабуванні систем або спробах адаптувати їх до складніших або нестандартних сценаріїв. Основна проблема полягає в залежності від хмарної інфраструктури: у разі збою чи відключення серверів зникає зв’язок між пристроями навіть на локальному рівні, що унеможливлює виконання автоматизацій, які мали б працювати незалежно від інтернет-з’єднання. Втрата доступу до хмари перетворює розумну систему на звичайний набір неузгоджених пристроїв, в яких втрачається логіка взаємодії, затримується або повністю зникає зворотний зв’язок, а сценарії автоматизації стають недоступними. Окрім технічної вразливості до зовнішніх факторів, обмеження також виявляються у недостатній гнучкості налаштувань — додатки на кшталт Smart Life надають лише базові шаблони, яких замало для складної логіки, що передбачає багатокомпонентні умови, наприклад, одночасне врахування часу доби, стану декількох пристроїв і зовнішніх подій [4].

У таких платформах відсутня можливість створення умовних конструкцій з вкладеними логічними перевірками, а також існує обмеження на кількість і тип тригерів, які можуть запускати сценарії, що значно звужує потенціал автоматизації. Обмеження стосуються і самого контролю над пристроями: наприклад, у випадку роботи з розумними розетками неможливо отримати дані про миттєве навантаження з високою точністю або здійснити керування живленням із затримкою в межах мілісекунд. Часто такі функції або недоступні зовсім, або приховані від користувача, а отже, не можуть бути інтегровані у більш складну логіку роботи системи.

1.2 Функціональна модель системи управління об’єктами в IoT-середовищі

Переходячи до розгляду функціональної моделі системи управління об’єктами в середовищі Інтернету речей, одразу стає очевидним, що така модель характеризується багаторівневою архітектурою, здатною охоплювати як фізичні пристрої, так і програмно-логічні компоненти, необхідні для збору, обробки, передавання та регулювання інформації, що надходить від середовища. Ця архітектура формує структурну основу для взаємодії компонентів IoT-системи та дозволяє узгоджено функціонувати як автономним пристроям, так і централізованим чи децентралізованим механізмам управління. Аналізуючи структуру типової системи, типи пристроїв, які в ній використовуються, протоколи взаємодії між її елементами, а також особливості реалізації централізованого та децентралізованого підходів до керування, формується цілісне уявлення про функціональну модель системи управління в межах сучасного IoT-середовища [5-8].

Структура типової IoT-системи. Типова IoT-система умовно поділяється на чотири рівні, кожен з яких відповідає за конкретний етап функціонування системи. Першим є рівень спостереження, до якого належать фізичні пристрої, здатні здійснювати вимірювання параметрів навколишнього середовища або впливати на нього через певні дії. Серед таких пристроїв — сенсори температури, вологості, диму, витоку води, руху, а також актуатори, що дозволяють вмикати або вимикати лампи, розетки, електроклапани, реле тощо. Всі ці пристрої безпосередньо забезпечують збір первинної інформації та виконання фізичних команд, завдяки чому формуються базові одиниці даних і реагування. Другим є мережевий рівень, який виконує функцію транспортного каналу між фізичними пристроями та центральними вузлами обробки даних. Для забезпечення зв’язку використовуються різноманітні комунікаційні протоколи, серед яких локальні Wi-Fi, Bluetooth, Zigbee або Thread, а також глобальні канали зв’язку на кшталт Ethernet, LTE або

LoRaWAN. Вибір конкретного каналу залежить від сценарію розгортання, вимог до енергоспоживання, пропускної здатності та дальності передачі сигналу [5].

Третім компонентом є рівень обробки, до складу якого входять локальні контролери та шлюзи, які безпосередньо приймають дані від сенсорів, обробляють їх відповідно до заданих алгоритмів і видають керуючі сигнали актуаторам. У хмароорієнтованих системах цю функцію можуть виконувати віддалені сервери або платформи, що дає змогу зменшити вимоги до локального обладнання, хоча при цьому виникає залежність від зовнішньої інфраструктури. У локалізованих рішеннях логіка обробки реалізується безпосередньо у домашніх шлюзах або на одноплатних комп'ютерах, що підвищує автономність і стійкість до втрати зв'язку з інтернетом. Завершальним є рівень взаємодії з користувачем, що забезпечує зручний доступ до моніторингу, налаштування та управління системою за допомогою мобільних застосунків, веб-інтерфейсів або спеціалізованих панелей. На цьому рівні відображаються стани пристроїв, спрацьовування сценаріїв, графіки зміни параметрів, а також відбувається ручне втручання або коригування автоматизації за потреби (див. рис. 1.2) [5].

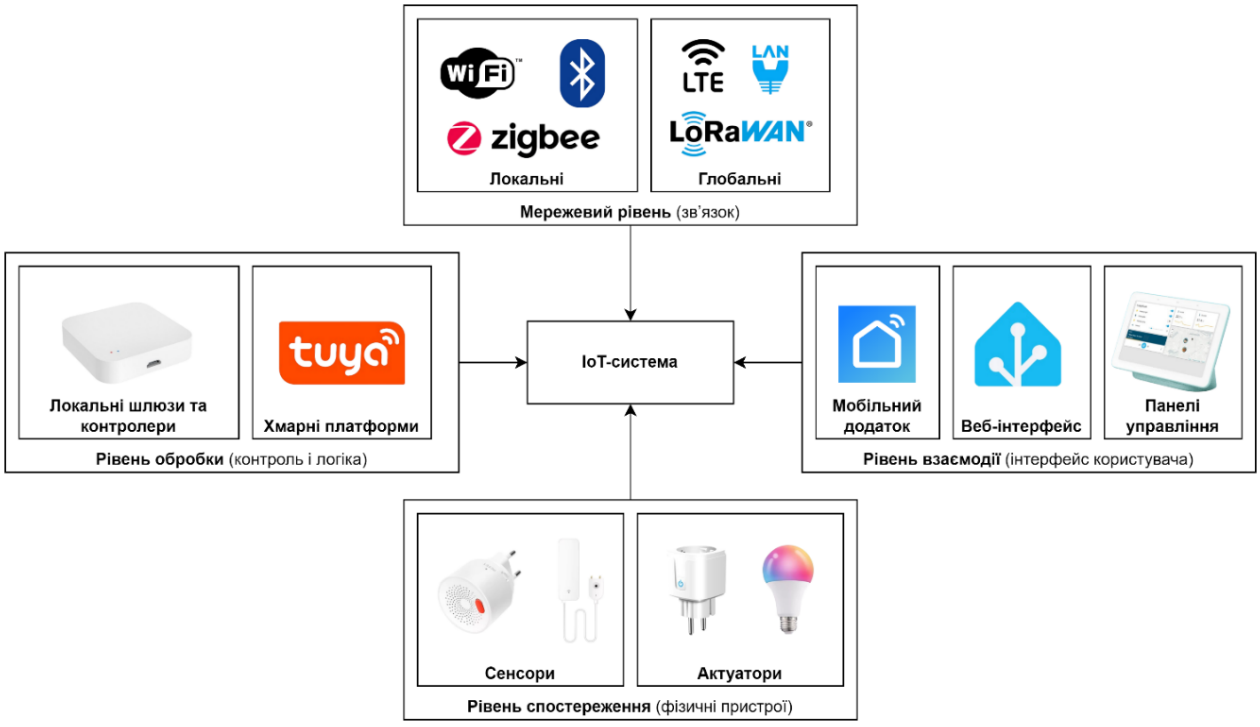

Рисунок 1.2 – Загальна структура IoT-системи управління

Функціональна модель як єдина структура вимагає тісної та безперервної взаємодії між усіма рівнями, оскільки втрата або порушення одного з елементів порушує цілісність системи в цілому. Надійність, масштабованість і стабільність зв'язку між рівнями стають критичними умовами для забезпечення безперервного функціонування системи, своєчасного реагування на зміни середовища та точного виконання алгоритмів автоматизації. Саме в цій взаємодії виявляється сутність IoT як інфраструктури, здатної поєднувати фізичні процеси з цифровим керуванням, формуючи гнучке, адаптивне й інтелектуальне середовище.

Типи керованих пристроїв. У середовищі Інтернету речей формується досить широка номенклатура керованих пристроїв, які в залежності від контексту застосування можуть бути інтегровані у побутові, офісні чи промислові інсталяції. Така різноманітність забезпечує побудову гнучких і масштабованих систем управління, в яких фізичні пристрої з різним функціональним призначенням працюють у складі єдиної логічної архітектури. Найпоширенішими прикладами є розумні розетки, що дозволяють дистанційно вмикати або вимикати електроприлади, здійснювати моніторинг енергоспоживання, а також задавати автоматичні сценарії та таймери для підключеного обладнання. Окрему категорію становлять системи освітлення з можливістю регулювання яскравості, кольору, ввімкнення за розкладом чи у відповідь на сигнали від сенсорів — усе це значно підвищує гнучкість автоматизації [6].

Водночас до критично важливих пристроїв належать сенсори витоку води та газу, які здатні сповіщати про аварійні ситуації та взаємодіяти зі сценаріями перекриття подачі відповідних ресурсів, якщо встановлені електромагнітні клапани. Подібно до них працюють детектори диму та чадного газу, які орієнтовані на вчасне виявлення загроз та запуск сигналізації або повідомлень до централізованої системи. Суттєве значення мають і кліматичні сенсори, зокрема датчики температури, вологості, атмосферного тиску, які дозволяють реалізувати логіку керування опаленням, кондиціонуванням, вентиляцією або зволоженням повітря в автоматичному режимі. Не менш функціональною є категорія пристроїв, що забезпечують керування старими побутовими приладами через інфрачервоний

розумний пульт, дозволяючи інтегрувати телевізори, кондиціонери або мультимедійні центри, які не підтримують сучасні протоколи зв'язку, у загальну систему автоматизації. Завдяки постійному розширенню асортименту і функцій IoT-пристроїв формується потенціал створення все більш складних сценаріїв, що охоплюють практично будь-який аспект взаємодії людини з простором [6].

Основні протоколи взаємодії. Функціонування таких пристроїв передбачає чітку координацію через протоколи взаємодії, які поділяються на мережеві та прикладні, забезпечуючи обмін даними між вузлами, синхронізацію та обробку команд. Найбільш розповсюдженим протоколом у побутовому середовищі є Wi-Fi, який завдяки простоті інтеграції з домашніми мережами забезпечує підключення пристроїв без потреби в додаткових шлюзах, хоча й поступається у питанні енергоефективності. Альтернативу становить Bluetooth, який частіше використовується в компактних пристроях-компаньйонах, здатних взаємодіяти зі смартфонами або планшетами. Значним у контексті енергоощадного підходу є протокол Zigbee, який дозволяє вибудовувати розгалужені мережі з низьким енергоспоживанням та динамічною топологією, хоча в даній реалізації він не застосовувався через наявність лише Wi-Fi-інфраструктури [7].

Особливої уваги заслуговує протокол MQTT, що забезпечує модель обміну повідомленнями «видавець–підписник» і активно використовується в системах реального часу, особливо у випадку локальних інтеграцій через платформи на кшталт Home Assistant. Він оптимізований для стабільної роботи навіть при обмежених ресурсах і добре підходить для автономних систем. У свою чергу, у випадку хмарно орієнтованої взаємодії застосовується власний протокол Tuya Cloud API, що дозволяє організувати керування пристроями через зовнішні застосунки, здійснювати автентифікацію, налаштовувати сценарії та контролювати стани пристроїв через інтернет. Хоча такий підхід полегшує початкову інтеграцію, він значно залежить від стабільного підключення до хмари, що може створювати обмеження в умовах автономного використання. У результаті кожен із описаних протоколів забезпечує різний рівень гнучкості, енергоефективності, автономності або залежності від зовнішніх сервісів, що зумовлює вибір конкретної технології

відповідно до задачі, наявної інфраструктури та вимог до безперервності керування пристроями у системі IoT [7].

Модель централізованого та децентралізованого управління. Архітектура керування в середовищі інтернету речей може формуватись за різними підходами, серед яких основними виступають наступні моделі:

— централізована;

— децентралізована

У централізованій моделі передбачається наявність одного головного вузла, такого як хмарний сервер або локальний контролер, який збирає інформацію від усіх підключених пристроїв, обробляє її та надсилає керуючі команди назад. Такий підхід здебільшого реалізується через комерційні платформи на зразок Tuya IoT Cloud, Xiaomi Mi Home, Amazon Alexa або інших подібних екосистем, які надають готові інструменти для швидкої інтеграції, зручного масштабування та підтримки широкого спектра пристроїв. Завдяки цьому значно спрощується старт, налаштування та щоденне використання системи навіть без глибокого технічного розуміння, однак залежність від зовнішнього сервера та стабільного інтернет-з'єднання, обмежена можливість повного контролю над даними та сценаріями, а також відсутність повної автономності часто стають критичними недоліками в середовищах з підвищеними вимогами до безпеки, надійності та гнучкості [8].

Натомість децентралізована модель пропонує альтернативну концепцію, у межах якої не існує єдиного центру обробки, а логіка управління реалізується або через локальні контролери, або безпосередньо через розподілену взаємодію між пристроями. У таких системах кожен пристрій здатен безпосередньо комунікувати з іншими учасниками мережі, обмінюватися даними та реагувати на події без залучення зовнішнього сервера, що значно підвищує рівень автономності та дозволяє гнучко адаптувати функціональність під конкретні потреби. Серед прикладів реалізації такої моделі особливо поширеним є програмне забезпечення Home Assistant, що забезпечує розширене конфігурування, локальне зберігання інформації, незалежність від хмари та високий рівень кастомізації. Однак використання подібних рішень передбачає початкову складність налаштування,

необхідність знань щодо мереж, протоколів та конфігурації обладнання, що може бути бар'єром у непідготовлених середовищах (див. рис. 1.3) [8].

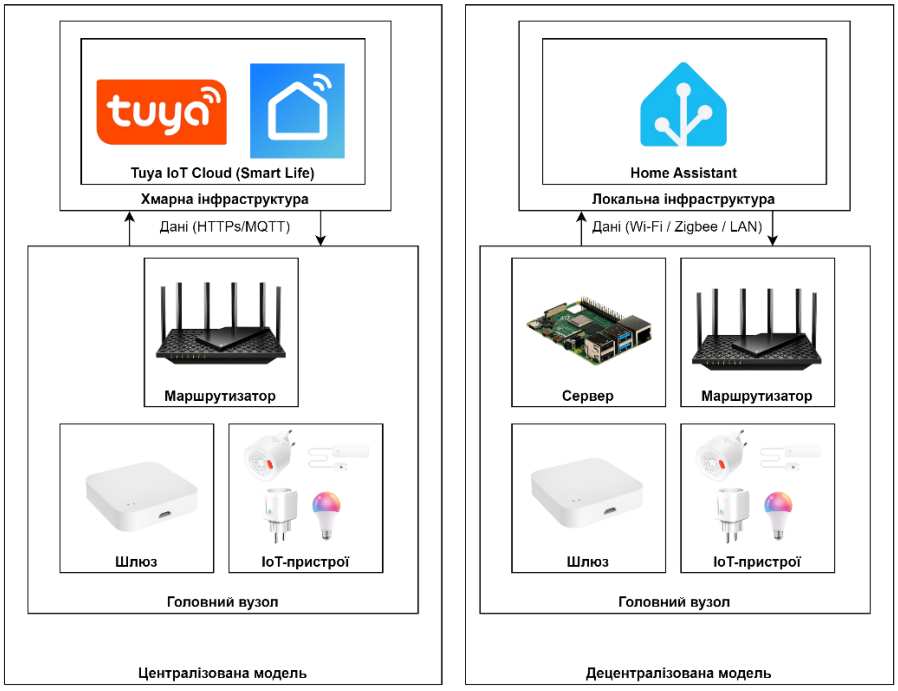

Рисунок 1.3 – Централізована та децентралізована моделі керування IoT

У деяких випадках практичним виявляється компромісний гібридний підхід, який поєднує переваги обох моделей. Наприклад, хмарні платформи можуть використовуватись для первинного налаштування, моніторингу або забезпечення доступу ззовні, тоді як локальні сервери забезпечують автономне виконання сценаріїв, обробку даних та взаємодію між пристроями без потреби в постійному зовнішньому з'єднанні. Такий підхід дозволяє досягти балансу між зручністю, стабільністю, безпекою та гнучкістю, що є особливо цінним у випадках використання IoT-систем у домашніх умовах, офісах або навіть промислових об'єктах, де різні зони можуть мати різний рівень критичності та вимоги до безперервності обслуговування [8].

Загалом функціональна модель системи інтернету речей має багаторівневу структуру, яка охоплює:

— апаратне забезпечення;

— мережеві протоколи;

— інтерфейси керування;

— сценарії взаємодії.

Розуміння принципів роботи пристроїв, специфіки протоколів і типу вибраної архітектури управління дозволяє проєктувати ефективні рішення для автоматизації та моніторингу в широкому спектрі середовищ. Остаточний вибір між централізованою, децентралізованою чи комбінованою моделлю визначається:

— специфікою поставлених завдань;

— вимогами до автономності;

— безпеки та адаптивності;

— доступною інфраструктурою.

З огляду на стрімкий розвиток галузі, дедалі більшої популярності набувають відкриті та гнучкі рішення, які не обмежуються єдиною платформою, дозволяють глибоку персоналізацію та зберігають незалежність системи від зовнішніх факторів.

1.3 Принципи побудови інформаційної взаємодії у хмарному середовищі

У середовищі Tuya, яке включає такі мобільні додатки як Smart Life або Tuya Smart, принцип побудови інформаційної взаємодії базується на централізованій архітектурі, де хмарна інфраструктура Tuya IoT Cloud виконує функції посередника між кінцевими пристроями та користувачем. Сам додаток Smart Life, який використовувався під час аналізу, є найбільш поширеним серед продуктів, побудованих на основі Tuya, хоча функціонально подібний до інших варіантів на цій платформі. Сама платформа Tuya IoT Cloud позиціонується як масштабована екосистема для підключення, моніторингу, управління та автоматизації мільйонів розумних пристроїв різного призначення — від освітлення до сенсорних датчиків, забезпечуючи стандартизовану модель взаємодії [9-12].

Взаємодія пристроїв із хмарним середовищем Tuya. Кожен сумісний пристрій, наприклад розумна розетка, лампа або сенсор, у процесі первинного налаштування підключається до локальної Wi-Fi мережі користувача, після чого проходить етап авторизації через мобільний додаток, який виступає посередником між пристроєм та хмарною платформою. Після цього встановлюється зашифроване з'єднання з Tuya IoT Cloud на основі протоколів MQTT або HTTPS, залежно від типу пристрою, що забезпечує передачу команд у захищеному вигляді. Усі подальші команди керування, такі як вмикання або вимикання пристрою, зміна параметрів, моніторинг стану, більше не відбуваються напряму між смартфоном і самим пристроєм, а передаються через хмарний сервер. Саме цей підхід дозволяє забезпечити віддалене управління з будь-якої точки, де доступний інтернет, оскільки вся логіка обробки запитів та відповіді на них відбувається у хмарному середовищі [9].

Така модель значно спрощує перші кроки користувача з інтеграцією системи, оскільки не потребує додаткового обладнання на кшталт локального сервера чи контролера, знімаючи потребу в технічній підготовці та складному налаштуванні. Вона також дозволяє легко масштабувати інфраструктуру, додавати нові пристрої, формувати автоматизації в інтерфейсі мобільного додатка без додаткового програмування. Водночас цей підхід створює повну залежність від хмарної платформи, оскільки без активного інтернет-з'єднання або у разі недоступності серверів Tuya пристрої втрачають здатність до взаємодії, навіть якщо перебувають у межах локальної мережі (див. рис. 1.4) [9].

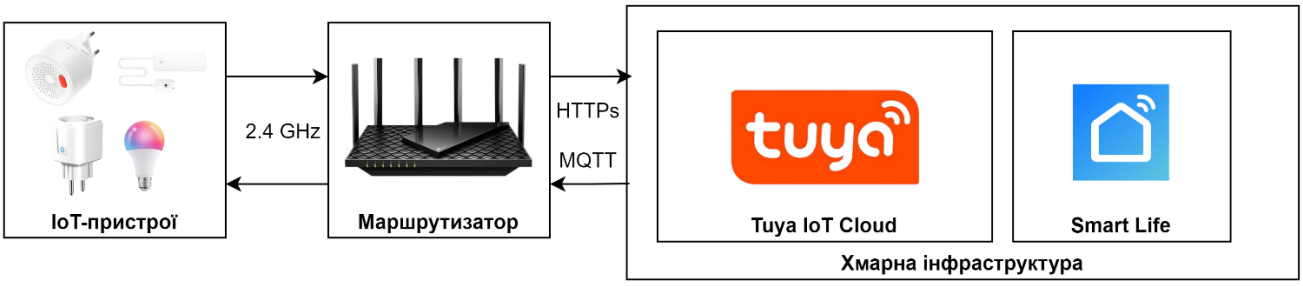

Рисунок 1.4 – Архітектура взаємодії пристроїв із хмарним середовищем Tuya

Таким чином, централізований принцип інформаційної взаємодії у Tuya IoT Cloud забезпечує простоту, доступність та віддалене керування, але вимагає постійної присутності стабільного з'єднання з хмарною інфраструктурою для коректної роботи всієї системи.

Робота мобільного застосунку Smart Life як інтерфейсу керування. У середовищі Tuya, яке охоплює платформи на кшталт Smart Life, основний принцип побудови інформаційної взаємодії базується на централізованій архітектурі, в якій усі пристрої користувача підключаються до хмарної інфраструктури Tuya через локальну Wi-Fi мережу. На прикладі мобільного додатку Smart Life, що є одним із найпоширеніших інтерфейсів управління цією екосистемою, можна простежити типову схему підключення, взаємодії та автоматизації. Після встановлення застосунку та проходження авторизації, користувач може додавати пристрої, які проходять процедуру реєстрації в хмарі, після чого починають обмінюватися даними з сервером Tuya через зашифровані протоколи MQTT або HTTPS. Усі команди з керування пристроєм — наприклад, увімкнення світла, перевірка стану розетки чи зміна яскравості — передаються не безпосередньо між додатком і пристроєм, а спочатку надсилаються на сервер Tuya, обробляються там і лише потім спрямовуються на відповідний пристрій (див. рис. 1.5) [10].

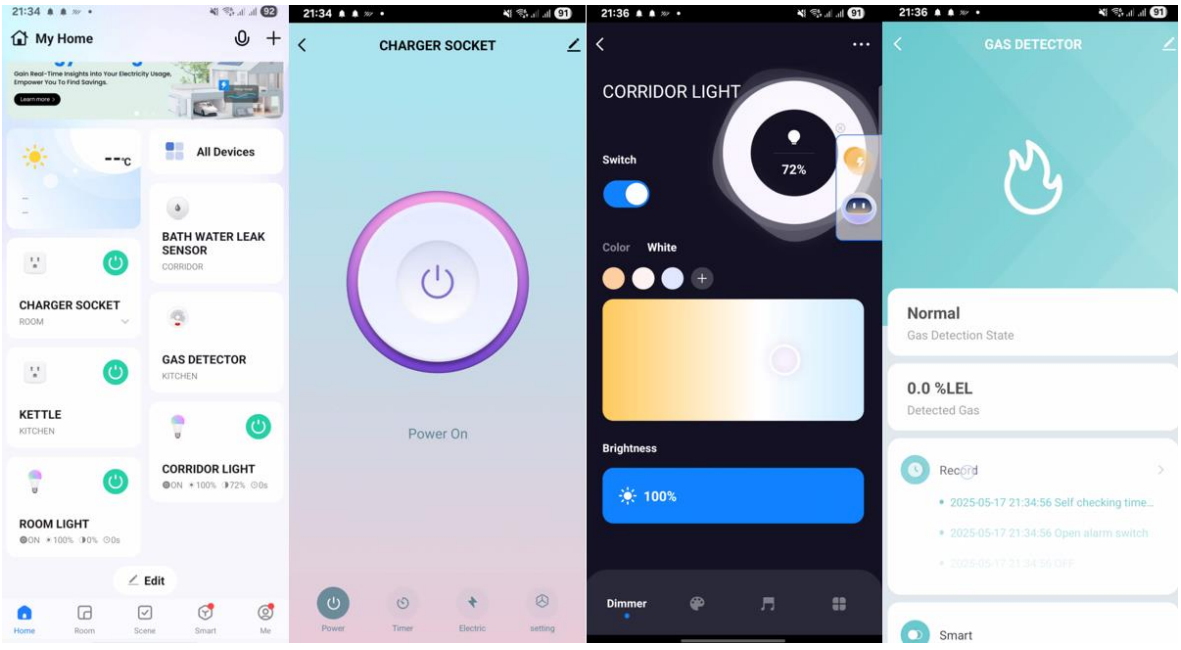

Рисунок 1.5 – Інтерфейс застосунку Smart Life для керування пристроями

Подібна модель дозволяє здійснювати контроль із будь-якої точки, де є доступ до інтернету, що значно підвищує зручність використання, особливо для початківців, оскільки не вимагає локальних серверів або складного налаштування. Разом із тим, вона повністю залежить від стабільності інтернет-з'єднання та доступності хмарної інфраструктури, адже при втраті зв'язку з сервером керування пристроями навіть у межах локальної мережі стає неможливим. Додаток Smart Life виконує роль головного інструмента взаємодії з екосистемою Tuya, забезпечуючи користувачеві:

— перегляд пристроїв;
— моніторинг їхнього стану в реальному часі;
— контроль параметрів;
— групування;
— налаштування сценаріїв та автоматизацій.

Передбачено також отримання сповіщень про події, зміну стану чи тривоги, а також можливість надання доступу іншим користувачам, що актуально для сімейного або багатокористувацького використання [10].

Крім традиційного управління через інтерфейс додатку, підтримується інтеграція з голосовими асистентами на зразок Google Assistant або Amazon Alexa, що відкриває можливість керувати пристроями голосом, проте всі ці функції також реалізуються через хмару, а отже, залишаються залежними від наявності інтернету та роботи віддалених серверів.

Принципи автоматизації за подіями, станами, сценаріями. Особливу цінність в екосистемі Tuya має підтримка автоматизацій, які дозволяють будувати складні поведінкові сценарії на базі подій, змін стану, розкладу чи зовнішніх умов без необхідності постійного втручання. Архітектура автоматизацій базується на принципі «якщо – то», де тригери можуть бути як фізичними подіями — спрацювання сенсора руху, відкриття дверей, натискання кнопки, — так і умовами, пов'язаними зі зміною стану пристрою, наприклад, вмиканням або вимиканням розетки, лампи чи вентилятора (див. рис. 1.6) [11].

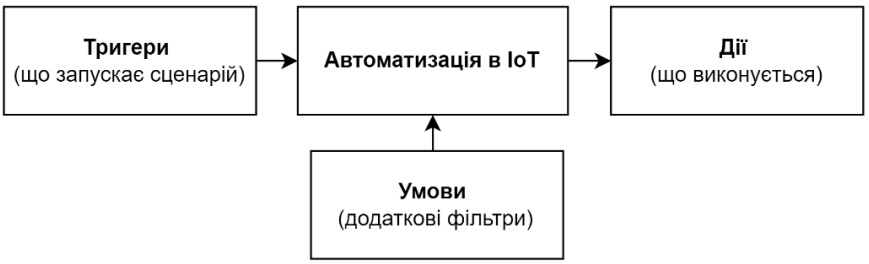

Рисунок 1.6 – Принцип автоматизації на основі подій та сценаріїв

Ще однією категорією тригерів виступають розклади, які дозволяють запускати дії за певним графіком, як-от щоденне вмикання світла о 7:00 ранку, а також зовнішні умови, до яких належать температура, вологість, геолокація чи погодні зміни. У відповідь на тригер може бути задана одна або кілька дій:

— увімкнення іншого пристрою;
— зміна режиму;
— надсилання повідомлення;
— активація раніше підготовленої сцени.

Наприклад, можна створити сценарій, що реагуватиме на виявлення руху у вітальні з 22:00 до 06:00 шляхом вмикання нічного освітлення та надсилання push-сповіщення на смартфон. Усі ці сценарії зберігаються в хмарному середовищі Tuya, а їхнє виконання, як і будь-яка інша взаємодія, напряму залежить від стабільності з’єднання з сервером. У разі його втрати жодна з автоматизацій не зможе бути виконана, навіть якщо всі пристрої фізично перебувають в одній локальній мережі, що є суттєвим обмеженням подібної архітектури, хоча й забезпечує високу гнучкість та мобільність керування у звичних умовах постійного інтернет-доступу.

Вразливості – затримки, залежність від Інтернету, втрати керованості. Хоч незважаючи на певну зручність і масштабованість, хмарна архітектура, на якій базується екосистема Tuya, має низку потенційних недоліків і вразливих місць, що проявляються як у побутовому використанні, так і в контексті довготривалої експлуатації. Однією з головних вразливостей є повна залежність від підключення до Інтернету: усі команди між користувачем і пристроєм передаються через зовнішні сервери, що фактично означає посередництво третьої сторони у кожній

дії. За відсутності інтернет-з’єднання або у випадку збою на стороні серверів Tuya керування пристроями стає неможливим. Винятком можуть бути лише ті пристрої, що мають локальні апаратні елементи керування, наприклад кнопку вмикання або перемикач, який безпосередньо вмикає або вимикає реле незалежно від наявності зв’язку. Це, наприклад, типово для розумних розеток, які дозволяють фізичне втручання, проте з телефону чи через додаток у такій ситуації керування вже недоступне, оскільки додаток не зможе побачити пристрій без підключення до хмари (див. рис. 1.7) [12].

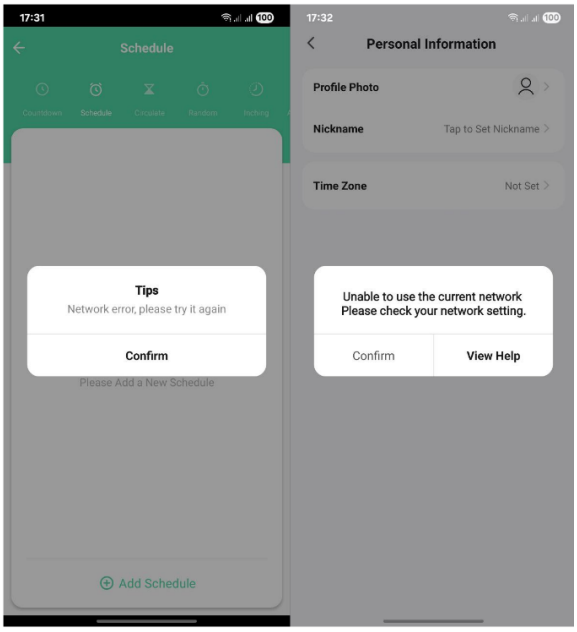

Рисунок 1.7 – Відсутність будь-якої взаємодії зі IoT пристроями

У разі наявності інтернету, але при поганій якості з’єднання або високому навантаженні на сервери, спостерігаються затримки у виконанні команд — від 1 до 5 секунд і більше. Для побутових потреб це може здатись дрібницею, але в критичних сценаріях, зокрема у системах безпеки або управлінні в реальному часі, навіть така затримка може бути неприйнятною. Така непередбачуваність і нестабільність є прямим наслідком архітектурної побудови, де кожна дія проходить через зовнішню інфраструктуру.

Окремо варто розглядати питання конфіденційності, оскільки вся інформація про стан пристроїв, історію їхньої роботи та сценарії автоматизацій зберігається на

сторонніх серверах. Користувач не має повного контролю над цими даними, що може викликати занепокоєння у тих, для кого приватність має принципове значення, включаючи приватних осіб, компанії та організації. Навіть у випадках, коли пристрій видаляється з облікового запису, а потім додається знову — чи до іншого облікового запису — сервер все одно продовжує зберігати історичну інформацію про його попередню активність, створюючи своєрідний «цифровий відбиток», що залишає пристрій частково прив’язаним до хмарної екосистеми незалежно від волі користувача [12].

Ще однією вразливістю є оновлення прошивок та зміни політик постачальника. У деяких випадках це може призводити до порушення сумісності пристроїв або навіть до припинення їхньої роботи зі сторонніми платформами, як це трапляється зі зміною інтерфейсів або функціоналу. Деякі моделі, хоч і зовні однакові, можуть мати різні прошивки, що веде до розбіжностей у поведінці або відсутності певних функцій, що створює додаткові труднощі при масштабуванні систем або інтеграції нових компонентів (див. рис. 1.8) [12].

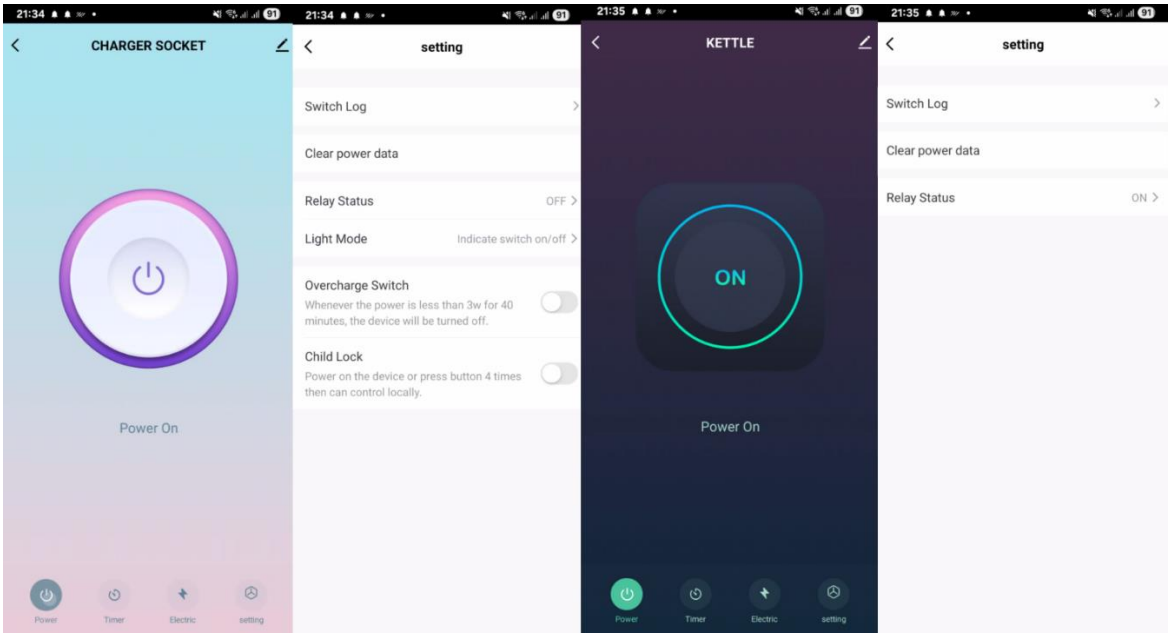

Рисунок 1.8 – Відмінність у інтерфейсі та функціоналі однакових IoT-пристроїв

Особливо критичною є відсутність можливості локального виконання автоматизацій, оскільки всі сценарії зберігаються і виконуються виключно у

хмарному середовищі. У випадку втрати інтернет-зв’язку навіть елементарна дія на кшталт увімкнення освітлення при відкритті дверей не буде виконана. Така централізована архітектура, що зосереджує всі логічні процеси на стороні постачальника послуг, виключає автономність, що для багатьох сценаріїв є неприйнятним обмеженням (див. рис. 1.9).

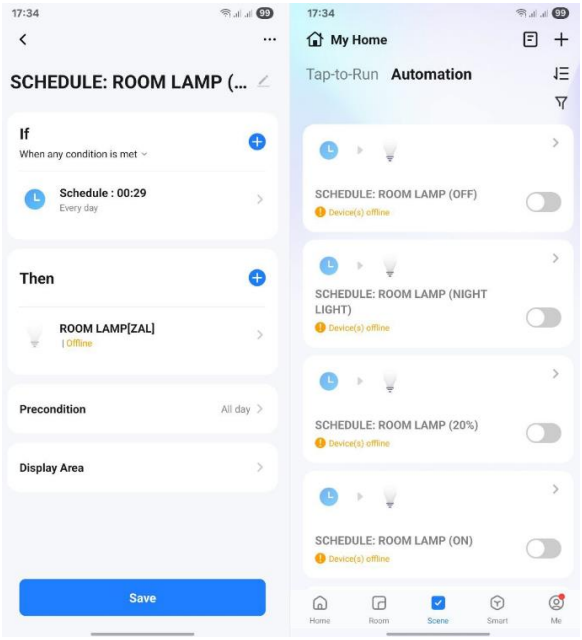

Рисунок 1.9 – Відсутність виконання налаштованих автоматизації у разі втрати інтернет-зв’язку

Узагальнюючи, архітектура хмарної інфраструктури Tuya разом із додатками Smart Life або Tuya Smart дійсно дозволяє створити масштабовану, функціональну та відносно просту у впровадженні екосистему керування пристроями. Завдяки централізованому підходу забезпечується сумісність із великою кількістю пристроїв, швидкий старт і мінімальні вимоги до знань. Водночас така модель супроводжується серйозними обмеженнями, пов’язаними зі:

— залежністю від інтернету;
— затримками у виконанні команд;
— відсутністю автономного керування;
— ризиками втрати конфіденційності.

1.4 Архітектура існуючої системи управління побутовими пристроями

У межах реалізованої системи керування побутовими пристроями, що функціонує через платформу Tuya IoT Cloud та хмарний застосунок Smart Life, вже впроваджено базову архітектуру керування об’єктами інтернету речей із фокусом на автоматизацію типових побутових процесів, підвищення рівня безпеки житла та енергоефективність. Уся інфраструктура побудована на основі Wi-Fi з використанням протоколу 2.4 ГГц, через який здійснюється синхронізація пристроїв із хмарою після їхнього додавання у мобільний застосунок. Ця архітектура охоплює набір пристроїв, які вже були фізично встановлені та протестовані в реальних умовах, зокрема (див. рис. 1.10):

— дві розумні розетки;
— дві розумні лампи;
— сенсор витоку води;
— датчик витоку газу.

Рисунок 1.10 – Зовнішній вигляд наявних керованих пристроїв

Незважаючи на обмежену кількість пристроїв, наявна конфігурація дозволяє реалізувати централізовану логіку автоматизації житлового середовища [13-16].

Типи пристроїв у наявній конфігурації. У першому випадку йдеться про дві розумні розетки типу Smart Wi-Fi Socket, які підтримують як базові функції керування електроживленням, так і моніторинг енергоспоживання – кожна з них налаштована з урахуванням специфіки підключених до неї пристроїв [13].

Перша використовується в спальні для заряджання мобільних телефонів, носимих пристроїв або інших побутових ґаджетів. Завдяки функції енергомоніторингу, розетка автоматично вимикається, коли потужність споживання знижується нижче заданого порогу (наприклад, 3 Вт), що дозволяє уникати зайвого споживання енергії та зменшити знос акумуляторів. Друга розетка встановлена на кухні та підключена до звичайного електрочайника, що не має вбудованих інтелектуальних функцій. У цьому випадку створено просту, але зручну автоматизацію: ввечері в чайник заливається вода, вмикається кнопка на корпусі, а у застосунку Smart Life для розетки встановлюється таймер увімкнення — наприклад, щодня о 6:30 ранку. Коли настає вказаний час, розетка подає живлення на чайник, який автоматично починає кип’ятити воду, забезпечуючи готовність до сніданку без додаткових дій користувача (див. рис. 1.11).

Рисунок 1.11 – Наявні IoT пристрої: Wi-Fi Smart Plug

Наступний тип пристроїв представлений двома інтелектуальними лампами, які можуть змінювати яскравість, колірну температуру (від теплого до холодного світла), а також підтримують RGB та CCT підсвітку. Кожна з ламп встановлена в

окремій кімнаті — одна у спальні, інша в передпокої — і може працювати за заданим розкладом. Наприклад, у вечірні години, починаючи з 20:00, лампа автоматично вмикається, забезпечуючи комфортне освітлення. Крім того, завдяки підтримці групового керування, лампи можуть об’єднуватись за зонами (кімнатами), що спрощує керування сценаріями. В даному випадку пристрої залишаються розділеними за локаціями, отже, кожна лампа функціонує незалежно у межах власної кімнати. У застосунку Smart Life передбачено можливість інтегрувати ці лампи в сценарії, що запускаються за часом або подією, наприклад, включення під час заходу сонця або при виявленні руху, хоча в цій конфігурації реалізовано лише базову автоматизацію за розкладом (див. рис. 1.12) [13].

Рисунок 1.12 – Наявні IoT пристрої: Wi-Fi Smart Bulb

Окрему категорію пристроїв становлять сенсори безпеки, серед яких у наявності є один сенсор витоку води, встановлений у санвузлі поруч із пральною машиною. Він постійно контролює наявність вологи в зоні підлоги та у випадку виявлення витоку миттєво надсилає сигнал до хмарної платформи Tuya, яка далі ініціює сповіщення у мобільному застосунку, що дозволяє користувачу оперативно реагувати на потенційні аварії, уникнувши затоплення. Сигнал передається як у вигляді push-сповіщення, так і через візуальну індикацію в інтерфейсі Smart Life (див. рис. 1.13) [13].

Рисунок 1.13 – Наявні IoT пристрої: Water Leak Sensor

Ще один критично важливий компонент архітектури — детектор витоку газу, розміщений на кухні. Пристрій постійно вимірює концентрацію метану в повітрі, та у разі перевищення порогу (наприклад, понад 5 одиниць у відповідній шкалі) активує вбудований гучний звуковий сигнал, а також надсилає тривожне повідомлення до хмари Tuya і відповідно — на смартфон користувача. Завдяки поєднанню локальної звукової індикації та хмарних сповіщень, забезпечується своєчасне попередження про потенційну небезпеку навіть у випадку відсутності інтернет-зв’язку або коли користувач не перебуває вдома (див. рис. 1.14) [13].

Рисунок 1.14 – Наявні IoT пристрої: Gas Leak Sensor

Сценарії роботи через застосунок Smart Life. У системах розумного дому, що базуються на екосистемі Tuya, мобільний застосунок Smart Life фактично виконує роль єдиного інтерфейсу доступу до всієї інфраструктури керування пристроями. Така централізована модель взаємодії є типовою для подібних рішень,

оскільки контроль здійснюється переважно через смартфони або планшети. У межах цього застосунку реалізовано низку базових функцій, зокрема можливість дистанційного керування пристроями у реальному часі. Наприклад, можна вмикати або вимикати розумні розетки, освітлення, а також отримувати миттєві сповіщення від сенсорів витоку води, диму або газу. Подібна інтеграція дозволяє не лише виявити загрозу, але й автоматично зреагувати на неї: наприклад, за наявності відповідного виконавчого пристрою, такого як електромагнітний клапан, система здатна автоматично перекрити подачу води або газу. У найпростішому випадку, коли такі пристрої не підключені, надходить лише повідомлення, однак навіть це дозволяє користувачу оперативно зреагувати на потенційну загрозу. Якщо, наприклад, у користувача є розумна розетка, до якої підключено пральну машину, і система фіксує витік води, то можна налаштувати автоматичне її вимкнення, і хоча таке рішення не завжди технічно виправдане, воно демонструє потенціал застосунку для побудови базових сценаріїв автоматизованого реагування без складної ручної конфігурації [14].

Окрім сценаріїв реагування на події, застосунок підтримує можливість створення розкладів і таймерів для автоматичного керування пристроями. Наприклад, освітлення або побутові прилади можуть бути запрограмовані на вмикання чи вимикання у певний час доби, що корисно для імітації присутності під час відпустки, підвищення енергоефективності або просто для зручності побуту, коли, скажімо, зранку електрочайник уже закипів. Усе це можна реалізувати безпосередньо зі смартфона, незалежно від місця перебування користувача, що забезпечується завдяки хмарній архітектурі Tuya. Також є можливість об’єднання пристроїв у групи — наприклад, усі лампи однієї кімнати можуть бути згруповані для одночасного керування. У розглянутій практичній реалізації така функція не застосовувалась, оскільки задіяні пристрої розташовані в різних кімнатах і не потребували синхронізованого керування. Сповіщення про події або загрози, що надходять у вигляді push-повідомлень, працюють навіть за вимкненого застосунку, забезпечуючи постійну поінформованість про стан системи (див. рис. 1.15).

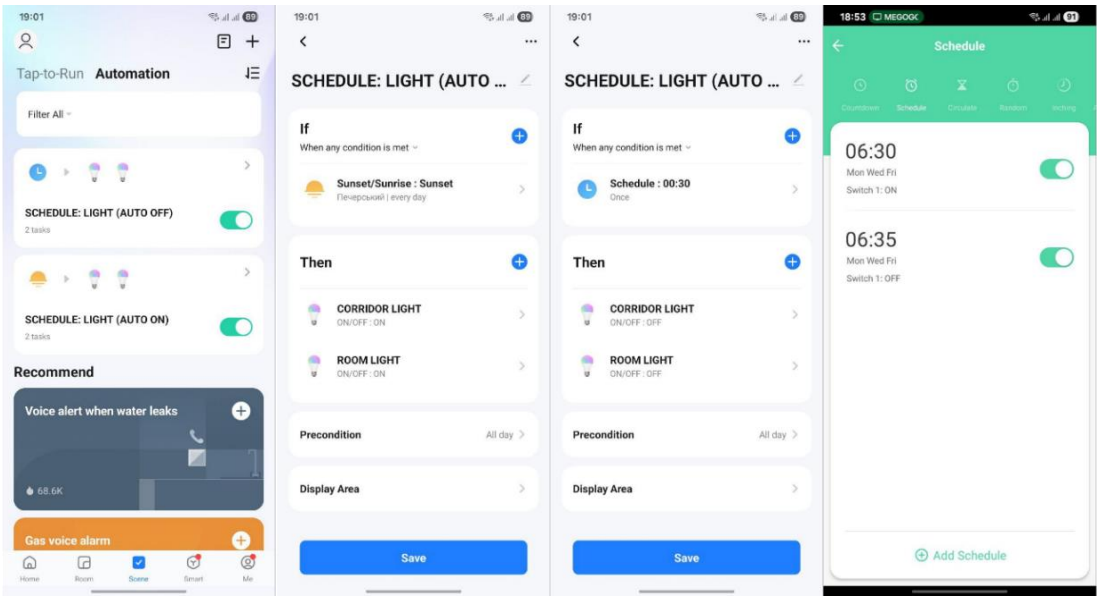

Рисунок 1.15 – Приклад сценаріїв та таймерів у Smart Life

Усе це разом формує інтуїтивно зрозумілий і функціонально повний інтерфейс для щоденного управління пристроями як локально, так і віддалено, без потреби в спеціальних навичках чи складних налаштуваннях [14].

Можливості та обмеження в автоматизації та групуванні. Що стосується можливостей та обмежень в автоматизації та групуванні пристроїв, система підтримує базовий набір функцій для побудови простих сценаріїв. Наприклад, при виявленні газу може бути налаштоване одночасне надсилання повідомлення і вимикання розеток у відповідному приміщенні. Також підтримуються події, прив’язані до часу — можна задати ввімкнення освітлення або інших пристроїв за фіксованим розкладом, скажімо, щодня, коли сідає сонце. Гнучкість налаштувань дозволяє одному сенсору слугувати тригером для кількох дій одночасно, що розширює варіанти взаємодії без ризику порушення логіки роботи інших сценаріїв. Водночас варто відзначити низку істотних обмежень. Платформа не дозволяє створювати складні умовні логіки з використанням конструкцій типу "if-else", що обмежує побудову сценаріїв із багатоступеневою перевіркою умов. Усі сценарії працюють виключно через хмарні сервіси, тому в разі відсутності підключення до інтернету вся логіка автоматизації припиняє функціонування. З цим також пов’язане обмеження щодо локального зберігання — сценарії не працюють

автономно без доступу до серверів Tuya. Ще одним обмеженням є логіка групування: можна об’єднувати лише пристрої одного типу, наприклад лише лампи або лише розетки, а створення змішаних груп, наприклад лампа та розетка, неможливе без сторонніх інтеграцій чи додаткового програмного забезпечення [15].

Попри ці обмеження, наявного функціоналу цілком достатньо для більшості побутових задач, оскільки типовому користувачу зазвичай не потрібна складна автоматизація або умовна логіка — базові функції спрацьовують надійно, забезпечуючи очікувану поведінку системи. Водночас для тих, хто прагне більшої гнучкості та локального керування, зазначені обмеження можуть стати причиною пошуку альтернативних платформ або розширення існуючої системи через сторонні інтеграції, що відкриває шлях до побудови більш складної, масштабованої та незалежної інфраструктури.

Загальна топологія системи. Якщо розглядати загальну топологію системи, реалізовану в межах цього проєкту, то вона побудована на основі класичної хмарно-орієнтованої архітектури, у якій усі пристрої підключаються до мережі Інтернет через домашній маршрутизатор та взаємодіють безпосередньо з хмарною платформою Tuya IoT Cloud (див. рис. 1.16).

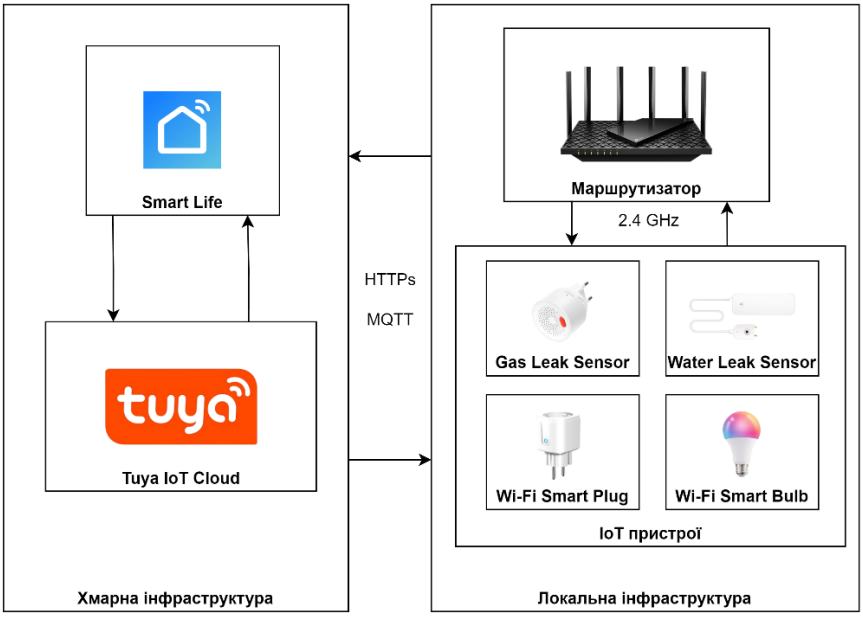

Рисунок 1.16 – Топологія поточної хмарної системи

У такій конфігурації передбачається, що розумні пристрої — зокрема дві лампи, дві розетки, детектор витоку газу та сенсор витоку води — комунікують із зовнішнім середовищем виключно за допомогою Wi-Fi-з’єднання, а всі зібрані або отримані дані одразу передаються до хмари, де централізовано обробляються, зберігаються, класифікуються та використовуються в межах налаштованих автоматизаційних сценаріїв. Після обробки ці події, оновлення станів або сповіщення спрямовуються до мобільного застосунку Smart Life, через який здійснюється інтерфейсне керування пристроями, перегляд історії подій та прийом повідомлень про виявлені аномалії [16].

У зворотному напрямку механізм дії функціонує аналогічно — при ініціюванні дії через застосунок відповідна команда надсилається до хмарного середовища Tuya IoT Cloud, яке вже потім транслює її до фізичного пристрою через інтернет і локальний маршрутизатор. Локальна мережа, таким чином, не виступає самодостатнім середовищем — пристрої не здатні взаємодіяти або реагувати на події автономно без зв’язку із зовнішнім сервером. Така архітектура дозволяє максимально спростити процес впровадження системи, не вимагаючи жодного додаткового обладнання типу локальних серверів або шлюзів, а також забезпечує доступність до керування системою з будь-якої точки світу, уникаючи потреби перебування в тій самій мережі, що й розумні пристрої, що також створює уніфіковане середовище для пристроїв різних виробників, за умови їхньої сумісності з платформою Tuya, яка забезпечує повну інтеграцію як у застосунок Smart Life, так і до альтернативного клієнта Tuya Smart — що фактично є його ідентичною версією [16].

Попри перелічені переваги, дана архітектура має низку критичних обмежень, зокрема повну залежність від стабільності інтернет-з’єднання. У разі його втрати автоматизація, сповіщення та керування можуть стати недоступними, оскільки всі логічні операції відбуваються у хмарному середовищі. Крім того, можливі затримки в обробці команд або навіть тимчасова втрата контролю над пристроями у випадках збою на серверах Tuya, що вже неодноразово фіксувалося на практиці [16].

Таблиця 1.1 – Характеристики пристроїв і сценаріїв у екосистемі Tuya (Smart Life)

№	Пристрій	К-ть	Локація	Основні функції	Приклади автоматизації
1	Розумна розетка (з енергомоніторингом)	2	Спальня, кухня	Вмик./вимик., моніторинг споживання	Автовимк. при <3 Вт (спальня), запуск чайника о 6:30 (кухня)
2	Інтелектуальна лампа (RGB+CCT)	2	Спальня, передпокій	Регулювання яскравості, кольору, температури світла	Увімкнення о 20:00, групове керування
3	Сенсор витоку води	1	Санвузол	Виявлення вологи, push-сповіщення	Сповіщення про витік, вимкнення розетки з пральною машиною
4	Сенсор витоку газу	1	Кухня	Виявлення метану, тривога, сповіщення	Push-сповіщення при витоку, опціональне вимкнення розетки
5	Застосунок Smart Life	—	Смартфон	Дистанційне керування, таймери, сповіщення	Розклад роботи пристроїв, реакція на сенсори
6	Хмарна платформа Tuya IoT Cloud	—	Онлайн-сервер	Обробка сценаріїв, зберігання, push-сповіщення	Усі автоматизації виконуються через хмару

У підсумку, наявна хмарна топологія, що реалізована у даній роботі (див. табл. 1.1), демонструє типовий підхід до побудови побутового середовища автоматизації, орієнтованого на мінімальні зусилля користувача при налаштуванні й використанні. Використання платформи Tuya IoT Cloud дозволяє досягти базового рівня функціональності, комфорту та безпеки, проте відсутність локальної автономності та обмежена логіка сценаріїв створює потребу у поступовому переході до локальних систем керування.

1.5 Причини переходу до відкритої системи

У межах побудованої інтелектуальної домашньої інфраструктури, що базується на моделі управління базовими об’єктами у мережі, значення має не лише вибір апаратної складової, а й типова архітектура програмної платформи, яка відповідає за керування пристроями, автоматизацію та взаємодію між компонентами. Обрана хмарна платформа Tuya IoT Cloud разом із застосунком Smart Life справді забезпечує зручність першого налаштування, швидкий запуск та простоту приєднання пристроїв, однак у процесі експлуатації поступово виявляється низка критичних обмежень, які неможливо усунути в рамках цієї замкненої екосистеми. Саме тому виникає об’єктивна необхідність у переході на більш відкрите середовище типу Home Assistant, де ключовим мотивом слугує прагнення до локального контролю, стабільності, автономності та розширеної гнучкості у кастомізації поведінки системи, відповідно до конкретних потреб та сценаріїв використання. Така зміна архітектури передбачає відмову від частини можливостей, зокрема від глобального керування через хмару, однак забезпечує стабільну взаємодію з пристроями всередині локальної мережі без залежності від інтернет-з’єднання. Встановлення локальної логіки автоматизації дає змогу не лише зберегти функціональність системи під час відключення зв’язку, а й сформувати сценарії поведінки, які технічно не реалізуються в межах стандартного

застосунку Smart Life, що особливо важливо в умовах, коли інтелектуальна інфраструктура має працювати безперервно, незалежно від зовнішніх факторів [17-20].

Обмеження в локальному управлінні та кастомізації у Smart Life. Незважаючи на свою поширеність серед масового користувача, платформа Tuya через застосунок Smart Life виявляє серйозні архітектурні обмеження, що унеможливлюють побудову справді стабільної, масштабованої та інтелектуально керованої системи. Насамперед ідеться про відсутність локального управління: уся логіка, автоматизація та обробка подій відбувається виключно на хмарних серверах Tuya, що означає повну залежність від зовнішнього інтернет-з'єднання навіть у ситуаціях, коли користувач і пристрої знаходяться в одній локальній мережі. У випадках перебоїв, зокрема під час типових для України блекаутів, система моментально втрачає функціональність, автоматизація перестає працювати, а користувач втрачає доступ до поточних і попередніх даних. У такій архітектурі навіть простий контроль ламп або розеток стає недоступним без хмари, що суперечить базовим вимогам до надійної локальної інфраструктури.

Додатково обмеження стосуються і самого конструктора сценаріїв — доступний лише мінімальний набір умов типу «якщо – тоді», без підтримки вкладених умов, комбінацій, логічних зв'язків чи залежностей між кількома подіями, що не дозволяє реалізовувати більш складну поведінку системи навіть за наявності відповідного обладнання. Відсутність журналу подій та засобів діагностики також позбавляє користувача змоги аналізувати збої чи поведінку окремих пристроїв: графіки присутні, але вони не містять деталізації, не фіксують логів, не дозволяють відстежити, чому саме сталася певна подія або помилка. Усе це знижує прозорість роботи та унеможливлює належний аналіз під час налагодження чи експлуатації [17].

Крім цього, інтерфейс Smart Life є уніфікованим, без можливості змінити структуру панелі керування, додати нові блоки даних, елементи візуалізації або створити індивідуальний вигляд дашборду під конкретні потреби. Попри стандартну зовнішню простоту, цей інтерфейс ускладнює користування через ще

одну технічну особливість: різні прошивки навіть одного й того самого пристрою (наприклад, розетки або лампи) можуть мати суттєво відмінний вигляд інтерфейсу або набір доступних функцій, що робить загальну взаємодію з системою фрагментарною й непослідовною.

З урахуванням усіх вищезазначених факторів, стає очевидно, що хоч платформа Tuya і Smart Life добре підходить для швидкого початку використання та базового віддаленого керування, вона не відповідає вимогам користувача, який прагне мати глибший контроль, гнучку логіку та стабільність системи. Платформа закритого типу без локального API, без прозорої діагностики, з обмеженою логікою сценаріїв та уніфікованим інтерфейсом фактично не здатна забезпечити реалізацію повноцінного інтелектуального середовища без критичних компромісів, що й обумовлює необхідність переходу на відкриту локальну платформу типу Home Assistant у межах подальшого розвитку цієї системи [17].

Мотивація до міграції та потреба у доступі без Інтернету. Однією з основних причин переходу до локальної інтеграції системи управління IoT-пристроями стала потреба в незалежному керуванні, що не вимагає постійного з’єднання з Інтернетом. У цьому контексті розглядається саме Home Assistant, який забезпечує таку можливість. Саме така локальна взаємодія дозволяє досягти більшої гнучкості, стабільності та безпеки, оскільки вся функціональність — від перемикання живлення на розетках до зміни яскравості освітлення чи зчитування даних із сенсорів — реалізується напряму, без посередництва хмарної інфраструктури. Крім основних можливостей керування, зберігається повна працездатність системи навіть у разі повної втрати інтернет-з’єднання. Автоматизації, сценарії, стани, дашборди та інші компоненти залишаються доступними, оскільки зберігаються на локальному сервері Home Assistant, а не у хмарному середовищі. Завдяки цьому забезпечується і конфіденційність, оскільки всі дані залишаються виключно в межах локальної мережі, і лише користувач вирішує, які дані збираються, як довго зберігаються та яким чином обробляються [18].

Окрему увагу заслуговує можливість розширення функціоналу завдяки відкритому коду — якщо певний пристрій не має офіційної підтримки, його все одно можна інтегрувати вручну або через кастомні рішення, чого неможливо досягти в межах закритої хмарної платформи, як-от Tuya IoT Cloud, що робить Home Assistant своєрідним мостом між закритими комерційними екосистемами та відкритим середовищем гнучкої кастомізації. При цьому вся наявна інфраструктура залишається без змін — ті самі Wi-Fi-пристрої Tuya можуть бути використані, але із повною локальною автономією завдяки зміні рівня контролю, а не апаратного забезпечення [18].

Затримки при виконанні сценаріїв та обмеження тригерів у хмарній системі. Важливим аргументом на користь локальної інтеграції виступає також наявність затримок у хмарній системі, які виникають при виконанні сценаріїв у Smart Life (Tuya IoT Cloud). Такі затримки коливаються в середньому від однієї до п’яти секунд, а в умовах нестабільного інтернету можуть бути ще більшими або навіть призводити до повного ігнорування тригера, що критично впливає на сценарії негайного реагування — наприклад, у випадку витоку води сенсор може миттєво зафіксувати подію, але сигнал має пройти через хмару, і лише потім надійти на смартфон. У разі поганого з’єднання повідомлення може запізнитися або не надійти взагалі, що зводить ефективність такої автоматизації нанівець [18].

Крім затримок, хмарна платформа має суттєві обмеження в логіці автоматизацій. Вона не дозволяє створювати складні сценарії, які враховують комбінацію кількох умов — наприклад, одночасне спрацювання декількох сенсорів із врахуванням часу доби, статусу інших пристроїв чи зовнішніх подій. Відсутність логічних операторів, перевірки станів, зворотного контролю чи умовних гілок значно обмежує можливості сценаріїв і робить систему непридатною для реалізації складних автоматизацій. Навпаки, в Home Assistant автоматизації можуть бути побудовані на будь-якій кількості умов, затримок, станів, операторів «і», «або», «не», а також враховувати зовнішні дані, що дозволяє створити точні та надійні сценарії, які реагують на події в реальному часі без сторонніх залежностей [18-19].

Потреба у стабільності, автономності та масштабуванні системи. Потреба у стабільності, автономності та масштабуванні системи прямо вказує на необхідність переходу до відкритої платформи, оскільки саме це дозволяє сформувати довготривале та гнучке середовище, яке не обмежується рамками одного постачальника або хмарного сервісу. У традиційній реалізації, заснованій на закритій інфраструктурі Tuya, основна залежність виникає від стану зовнішніх серверів, що в разі збою або недоступності повністю блокує керування пристроями. Натомість при локальному підході управління продовжується без жодних обмежень навіть за повної відсутності Інтернету, оскільки всі обчислення, обробка тригерів і реалізація сценаріїв відбуваються безпосередньо на внутрішньому сервері [19].

Особливо важливою така незалежність стає при розгляді критичних сценаріїв, наприклад, детекції витоку газу чи води, де навіть незначна затримка може мати реальні наслідки. Повна автономність у такому випадку не лише бажана, а й необхідна, адже дозволяє зреагувати миттєво, не чекаючи відповіді від хмари. Home Assistant реалізує подібний сценарій завдяки локальному виконанню автоматизацій, а в разі потреби забезпечує відправку повідомлень без додаткової оплати чи залежності від сторонніх сервісів. Навіть якщо з самого початку система не передбачає віддалене керування, завдяки підтримці VPN-тунелів та двофакторної автентифікації з’являється можливість організувати безпечний доступ ззовні без втрати контролю та без передачі даних стороннім сервісам [19].

У питанні масштабованості відкриті рішення мають ще одну суттєву перевагу: підтримка широкого спектра протоколів і пристроїв. На відміну від закритих екосистем, де розширення можливе лише в межах сумісного обладнання, у Home Assistant відкривається можливість додавати нові модулі, інтегрувати Bluetooth, Zigbee, MQTT або будь-які інші протоколи незалежно від виробника, що дозволяє не лише зберігати наявну інфраструктуру, а й розширювати її поступово, відповідно до потреб, без потреби заміни вже встановлених пристроїв [19-20].

РОЗДІЛ 2 АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ МОДЕЛІ СПЕЦІАЛІЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ

2.1 Аналіз керованих об’єктів і умов їх експлуатації

Для успішної реалізації моделі спеціалізованої системи управління об’єктами IoT необхідне глибоке розуміння структури керованих пристроїв, умов їх експлуатації та визначення основних завдань автоматизації, адже лише через комплексний аналіз можна забезпечити її ефективне впровадження у гнучку, масштабовану відкриту екосистему на кшталт Home Assistant. У межах такого підходу важливо не просто інтегрувати пристрої, а зрозуміти, яку функціональність вони можуть виконувати в реальних умовах повсякденного використання, і яким чином вони сприятимуть побудові повноцінного інтелектуального середовища, що має потенціал до розширення і адаптації, на відміну від обмеженої централізованої інфраструктури хмарного типу Tuya [19-23].

Визначення цілей автоматизації. Ключовим етапом тут є аналіз складу системи, її пристроїв, їхньої ролі в побуті та, що найважливіше, визначення сценаріїв автоматизації, які вже використовуються або можуть бути впроваджені. У фокусі такого аналізу опиняються реальні приклади автоматизації, орієнтовані на досягнення трьох головних цілей:

— підвищення енергоефективності;

— забезпечення безпеки;

— покращення зручності користування.

Перший напрям передбачає створення умов для оптимізації електроспоживання за рахунок автоматичного керування навантаженнями. У цьому контексті управління розетками, які вимикаються за умови відсутності активного споживання, є лише базовим прикладом. Наприклад, у хмарній системі Tuya вимкнення відбувається лише при зниженні споживання нижче трьох ват і з затримкою в кілька хвилин,

тоді як у Home Assistant можна додатково враховувати інші параметри: присутність користувача в будинку, час доби, геолокацію, рівень освітлення та інші фактори, що робить автоматизацію значно гнучкішою. Освітлення та електропостачання побутових приладів можуть підлаштовуватись не тільки під індивідуальні звички, а й під зміни умов навколишнього середовища [19].

Окремої уваги потребує блок, присвячений безпеці. Інтеграція датчиків витоку води або газу дозволяє не лише фіксувати інциденти, а й автоматично реагувати на них, наприклад, через сповіщення або активацію виконавчих пристроїв. Такий підхід стає особливо цінним у разі критичних ситуацій, коли навіть кілька секунд затримки, властивої хмарним системам, можуть стати фатальними. Завдяки локальній обробці даних та сценаріїв на базі Home Assistant вдається повністю усунути залежність від зовнішніх серверів, що суттєво підвищує надійність і оперативність роботи системи [19].

Наступний напрям пов’язаний із підвищенням рівня комфорту та зручності користування. Автоматизація освітлення відповідно до графіка, рівня освітленості зовні або присутності мешканців дозволяє створити інтуїтивне середовище, що не потребує постійного ручного втручання. Розетки, які активуються лише у визначений час або при потребі, сценарії ввімкнення зарядних пристроїв чи побутової техніки — усе це сприяє оптимізації повсякденних процесів і робить систему не просто інструментом контролю, а логічним продовженням життєвого простору, який адаптується під користувача (див. рис. 2.1) [19].

Рисунок 2.1 – Цілі автоматизації у побутових сценаріях

У межах аналізованої моделі система управління на базі відкритої платформи не обмежується роллю дистанційного контролера, а перетворюється на інтелектуальне ядро, здатне реагувати на зовнішні умови та дії користувачів, формуючи більш ефективну, безпечну і комфортну екосистему в порівнянні з централізованими хмарними рішеннями, які часто не надають достатньої гнучкості в автоматизації, масштабуванні й адаптації до потреб реального побуту.

Типи наявних пристроїв. На основі вже згаданої попередньо наявної інфраструктури важливо ще раз систематизувати типи пристроїв, які безпосередньо задіяні у побудові локальної моделі управління домашніми IoT-об’єктами через платформу Home Assistant. Йдеться про перехід від централізованої хмарної взаємодії до автономної локальної архітектури, яка забезпечує швидшу реакцію, більшу безпеку, гнучкість у налаштуваннях та мінімізацію залежності від зовнішніх серверів [20].

У даній реалізації використано чотири типи Wi-Fi-пристроїв з екосистеми Tuya, що вже експлуатуються в реальному домогосподарстві тривалий час, були попередньо прив’язані до хмарного облікового запису, протестовані, відв’язані та знову інтегровані для локального контролю з повним збереженням історії їх активності, що також свідчить про безперервне накопичення даних хмарною платформою, навіть після повторної прив’язки. Саме це зумовлює рішення відмовитися від такої схеми на користь локальної, ізоляційної архітектури без передачі зайвих даних третім сторонам.

До списку пристроїв насамперед входять дві розумні розетки, одна з яких використовується для управління електроживленням чайника на кухні в ранкові години, інша — для живлення зарядних пристроїв у спальні. Основна функція таких розеток полягає в дистанційному керуванні живленням та зборі даних про споживання електроенергії. У межах нової локальної моделі вони збережуть своє функціональне призначення, однак отримають нові сценарії взаємодії, розширені можливості реагування та деталізовану автоматизацію.

Окрім розеток, застосовуються дві розумні лампи Moes, розміщені у вітальні та спальні. Вони підтримують базову автоматизацію, наприклад, ввімкнення у

вечірній час та вимкнення вночі, що реалізовано засобами хмарної платформи Smart Life через прості графіки. В умовах переходу на Home Assistant з'являється можливість гнучкого адаптивного освітлення, враховуючи не тільки час доби, але й зовнішні фактори, з об'єднанням ламп у сцени та групи, де сценарії освітлення реагують на події або присутність користувача в приміщенні.

Наступною категорією є сенсор витоку води, що розташований у ванній кімнаті поблизу пральної машини. Його завдання полягає у виявленні вологи й надсиланні повідомлень у разі аварії. У хмарній версії сповіщення надходили через мобільний додаток, тепер же Home Assistant дозволить реалізувати кастомізовані реакції: надсилання повідомлень, фіксацію подій, створення історії спрацювань, а також у разі наявності відповідного обладнання — автоматичне вимкнення живлення в зоні ризику. Все це відбуватиметься без участі хмарної платформи, що значно підвищує контроль над системою.

Аналогічно, у кухонному приміщенні встановлений датчик витоку побутового газу, який раніше також працював через хмару Tuya, проте тепер отримає локальну автоматизацію на базі Home Assistant. Він виконує критично важливу функцію безпеки, повідомляючи про небезпечну концентрацію газу в повітрі. На новій платформі реалізовуються гнучкіші сценарії дій:

— звукові попередження;

— надсилання повідомлень;

— можлива реакція інших пристроїв за ланцюгом.

Усі згадані пристрої використовують Wi-Fi для підключення до локальної мережі, що не потребує окремих хабів чи шлюзів на відміну від рішень на базі Zigbee. Такий підхід спрощує побудову системи, однак має і зворотний бік — велика кількість Wi-Fi-пристроїв може перевантажувати маршрутизатор. У поточній конфігурації додавання нових пристроїв намагаються уникати саме з цієї причини [20].

Недоліки централізованого хмарного керування. Після аналізу реалізованої системи, яка вже містить певний набір пристроїв із екосистеми Tuya, а також визначення наявних автоматизацій і сценаріїв, що були запроваджені на базі централізованого хмарного керування через платформу Smart Life, сформувався

чіткий перелік обмежень та проблем, які виникали в процесі її використання, і які унеможливлюють повноцінну реалізацію бажаних функцій. З одного боку, централізована модель керування, побудована на хмарній інфраструктурі Tuya IoT Cloud, справді забезпечує достатню зручність для віддаленого доступу до пристроїв, особливо у випадках, коли потрібно швидко налаштувати базові сценарії без встановлення додаткового програмного забезпечення чи зміни конфігурацій пристроїв. Проте така модель виявляється непридатною, коли мова йде про реалізацію складніших сценаріїв, інтеграцію з локальними системами або забезпечення повного контролю над логікою автоматизації [22].

Залежність від хмарного з'єднання проявляється у вигляді регулярних затримок під час виконання команд, особливо коли йдеться про реакцію на події в реальному часі або побудову умовних ланцюгів дій. Відсутність локального бекенду унеможливлює впровадження гнучкої логіки обробки подій, збору даних для подальшого аналізу чи керування пристроями незалежно від стабільності інтернет-з'єднання. Також спостерігається суттєве обмеження щодо побудови складніших сценаріїв, оскільки доступні лише базові умови й дії, яких недостатньо для розгортання динамічної адаптивної системи керування. У процесі експлуатації стало очевидним, що навіть після повного скидання пристрою до заводських налаштувань усі дані, включно з історією, залишаються на серверах Tuya, що вказує на централізоване зберігання й обробку інформації третьою стороною без можливості користувача контролювати або обмежити цей процес [22].

Такі фактори, як обмежена безпека даних, відсутність повноцінної автономності, постійна залежність від зовнішніх серверів та обмежена логіка сценаріїв, виявилися критичними при подальшій роботі з системою. З часом початкові переваги централізованої моделі, зокрема простота налаштування, стабільність при мінімальних навантаженнях та можливість базової автоматизації, починають поступатися накопиченим недолікам, які не дозволяють реалізувати розширені можливості, необхідні для більш гнучкої, розширюваної та безпечної домашньої автоматизації. Поступово вимальовується необхідність переходу до локальної інфраструктури керування, яка здатна усунути більшість з описаних

обмежень, надати повний контроль над логікою, забезпечити швидкодію і, що важливо, зберігати дані виключно локально, без передачі їх третім сторонам (див. табл. 2.1) [19, 22].

Таблиця 2.1 – Ключові особливості та недоліки централізованого хмарного керування на базі екосистеми Tuya

Категорія	**Характеристика**
Тип підключення	Wi-Fi 2.4 ГГц
Тип інфраструктури	Хмарна (Tuya IoT Cloud)
Локальна робота без Інтернету	Відсутня (усі сценарії працюють лише через хмару Tuya
Підтримка умовних сценаріїв	Обмежено: без if-else, лише базові тригери (подія → дія)
Можливість групування	Можна об’єднувати пристрої одного типу (наприклад, лише лампи), неможливо змішані групи
Гнучкість автоматизації	Часові тригери, реакція на сенсори, прості одноетапні дії
Сповіщення	Push-сповіщення, навіть за вимкненого застосунку

У цьому контексті стає очевидною доцільність переходу до відкритої платформи, зокрема Home Assistant, яка дозволяє розгорнути повноцінну локальну систему управління на основі вже наявних пристроїв з екосистеми Tuya, з урахуванням подальшого вдосконалення, розширення функціональності та повної незалежності від хмарних служб.

2.2 Архітектура кастомізованої системи керування

Перехід від хмарного керування до локального вимагає побудови більш гнучкої, безпечної та автономної архітектури системи управління розумними пристроями, що дозволяє уникнути залежності від зовнішніх серверів та зменшити затримки при виконанні автоматизацій. У межах дослідження реалізується кастомізована система на базі Home Assistant, яка здатна взаємодіяти з наявними пристроями екосистеми Tuya виключно через локальну мережу. Усі дані передаються виключно в межах локальної інфраструктури, що унеможливлює витік або обробку інформації зовнішніми постачальниками, завдяки чому з'являється можливість не лише автономного керування, а й реалізації складніших сценаріїв з урахуванням локальних умов, даних сенсорів та взаємодії з іншими системами. Такий підхід передбачає використання локального контролера, в ролі якого виступає Home Assistant, який дозволяє реалізовувати кастомні інтерфейси, локальні автоматизації, зберігання історичних даних та роботу без доступу до інтернету (див. рис. 2.2).

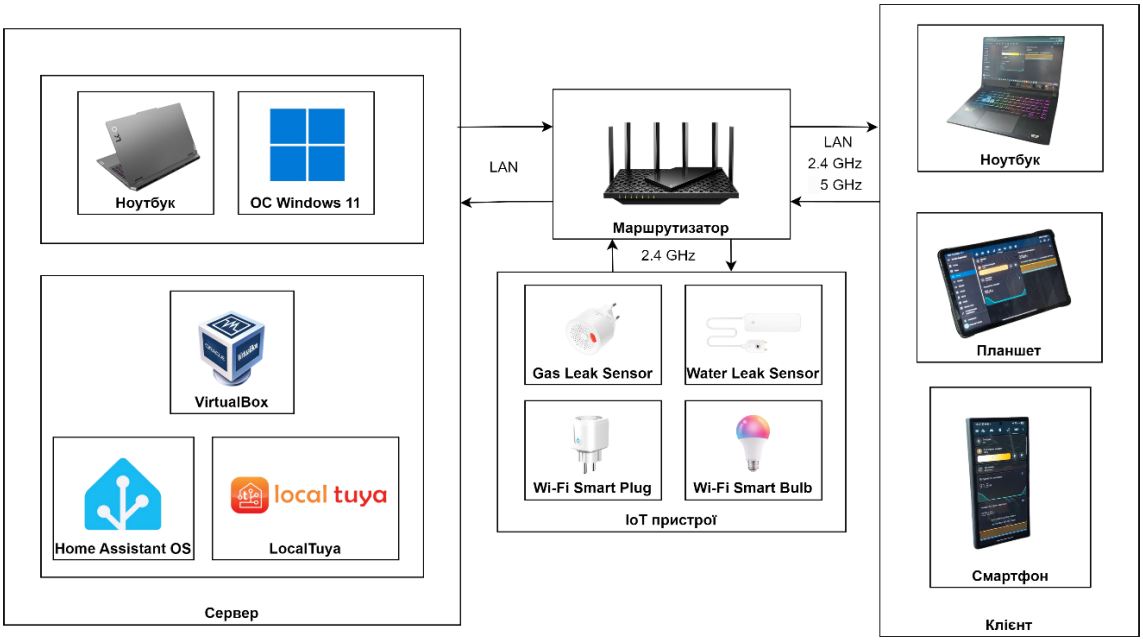

Рисунок 2.2 – Архітектура моделі спеціалізованої системи управління на базі Home Assistant

При цьому важливо звернути увагу на особливості встановлення та налаштування цього контролера, адже від правильності початкової конфігурації залежить стабільність і подальша масштабованість усієї системи [23-27, 35-36].

Підготовка системи – розгортання Home Assistant як локального контролера. Home Assistant — це операційна система з відкритим кодом, призначена для автоматизації розумного будинку, яка здатна працювати на широкому спектрі пристроїв, включно з одноплатними комп'ютерами (наприклад, Raspberry Pi, Orange Pi), міні-ПК (на кшталт Intel NUC) або будь-яким іншим пристроєм із підтримкою Linux. У рамках тестування обрана інсталяція Home Assistant OS на віртуальну машину, що дає змогу змоделювати повноцінне розгортання без використання додаткового фізичного обладнання. Віртуалізацію здійснено за допомогою VirtualBox, де використано офіційний образ Home Assistant у форматі VDI-диску. При створенні нової віртуальної машини обрано конфігурацію з 2 ГБ оперативної пам'яті та двома процесорними ядрами, а також налаштовано мережевий адаптер у режимі «Bridge», що забезпечує прямий доступ Home Assistant до локальної мережі як рівноправному учаснику поряд з іншими пристроями (див. рис. 2.3).

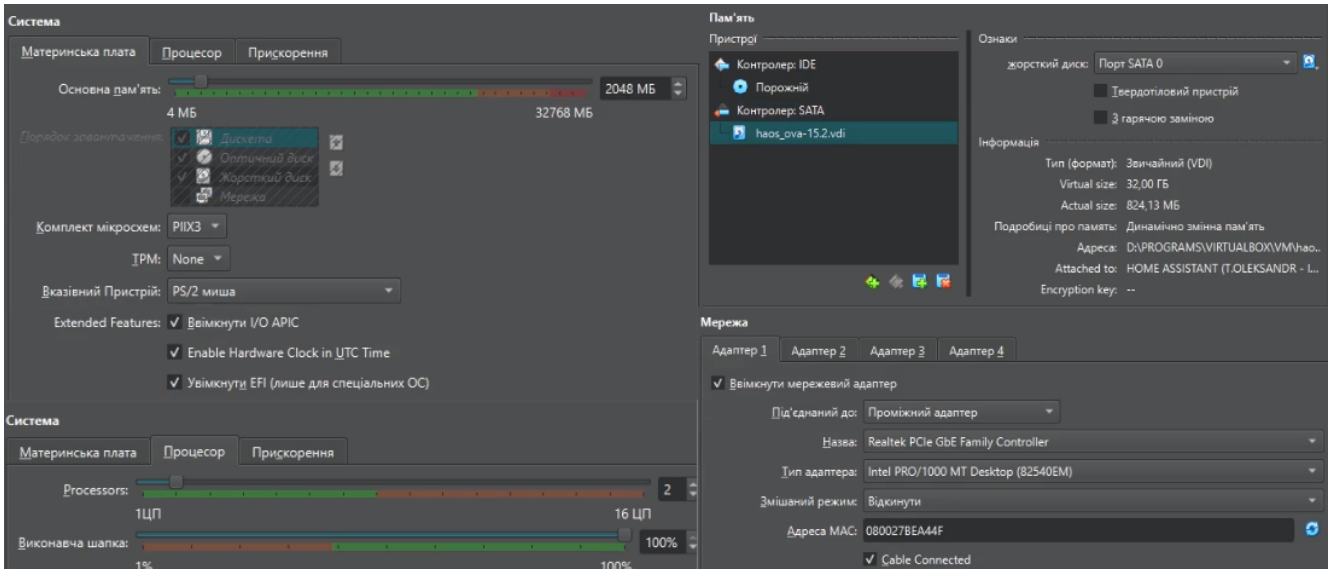

Рисунок 2.3 – Конфігурація віртуалізованого середовища VirtualBox

Додатково у налаштуваннях активовано завантаження в режимі EFI відповідно до вимог офіційної документації, без чого система не зможе коректно стартувати (див. табл. 2.2).

Таблиця 2.2 – Основні параметри розгортання Home Assistant у віртуальному середовищі

№	Характеристика	Значення / Пояснення
1	Тип контролера	Home Assistant OS
2	Спосіб розгортання	Віртуальна машина у VirtualBox
3	Тип образу	Офіційний VDI-диск Home Assistant
4	Оперативна пам’ять	2 ГБ
5	Кількість ядер CPU	2
6	Мережевий режим	Bridge (повноцінний доступ до локальної мережі)
7	Тип завантаження	EFI (обов’язковий для коректного запуску)
8	Платформа для тестування	VirtualBox (на хості з ОС Linux/Windows)
9	Пристрої, з якими сумісна система	Raspberry Pi, Orange Pi, Intel NUC, Linux-сумісні міні-ПК
10	Доступ до інтерфейсу	Через браузер за локальною IP-адресою після запуску

Після запуску віртуальної машини ініціюється процес завантаження Home Assistant, який займає кілька хвилин, після чого з’являється можливість підключення до веб-інтерфейсу локального контролера через браузер шляхом переходу за вказаною IP-адресою (див. рис. 2.4).

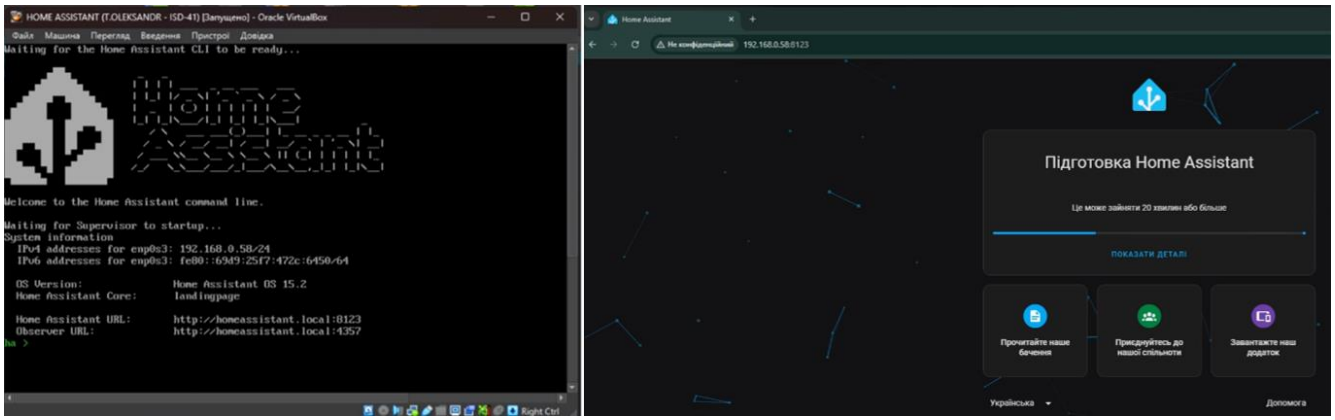

Рисунок 2.4 – Ініціалізація та підготовка Home Assistant OS

Підключення пристроїв через хмарну інтеграцію Tuya. Після завершення початкового налаштування Home Assistant у веб-інтерфейсі відкривається можливість створювати автоматизації та керувати основними параметрами середовища, однак на цьому етапі в системі ще відсутні підключені пристрої, тому першочерговим кроком стає реалізація базової інтеграції, яка дозволяє протестувати працездатність взаємодії між контролером та екосистемою пристроїв (див. рис. 2.5).

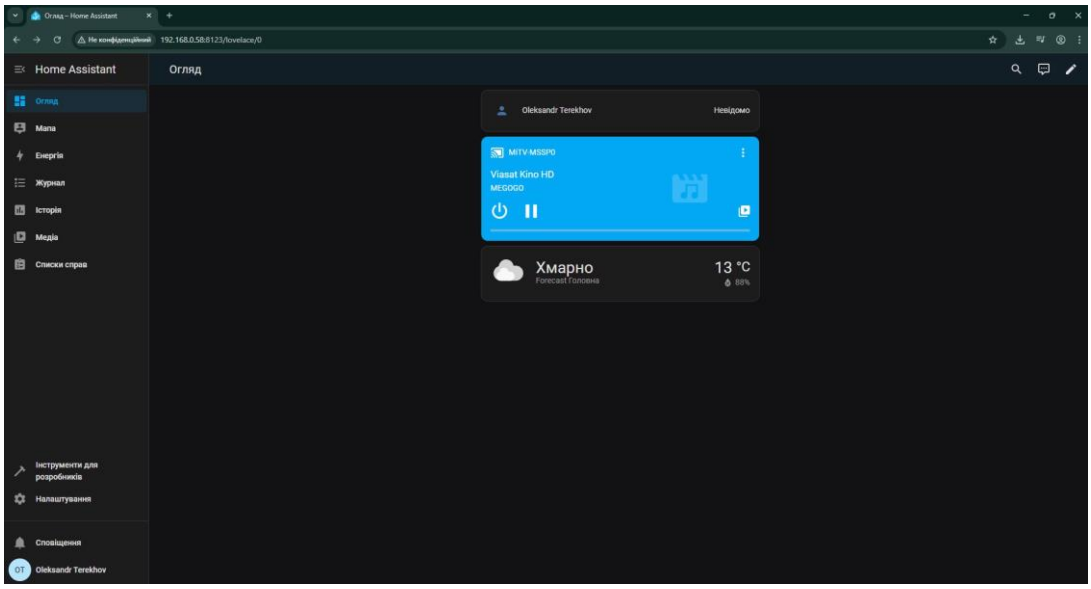

Рисунок 2.5 – Початковий дашборд у веб-інтерфейсі Home Assistant

Одним із найпростіших рішень для інтеграції пристроїв на початковому етапі є використання хмарного доповнення Tuya, яке вже присутнє у списку стандартних

інтеграцій Home Assistant і дозволяє швидко підключити до системи всю інфраструктуру пристроїв, що вже були попередньо додані у мобільний додаток Smart Life. Під час встановлення інтеграції Tuya, Home Assistant пропонує пройти процедуру авторизації, яка включає отримання спеціального коду у мобільному додатку, після чого потрібно відсканувати QR-код, що генерується в інтерфейсі Home Assistant (див. рис. 2.6).

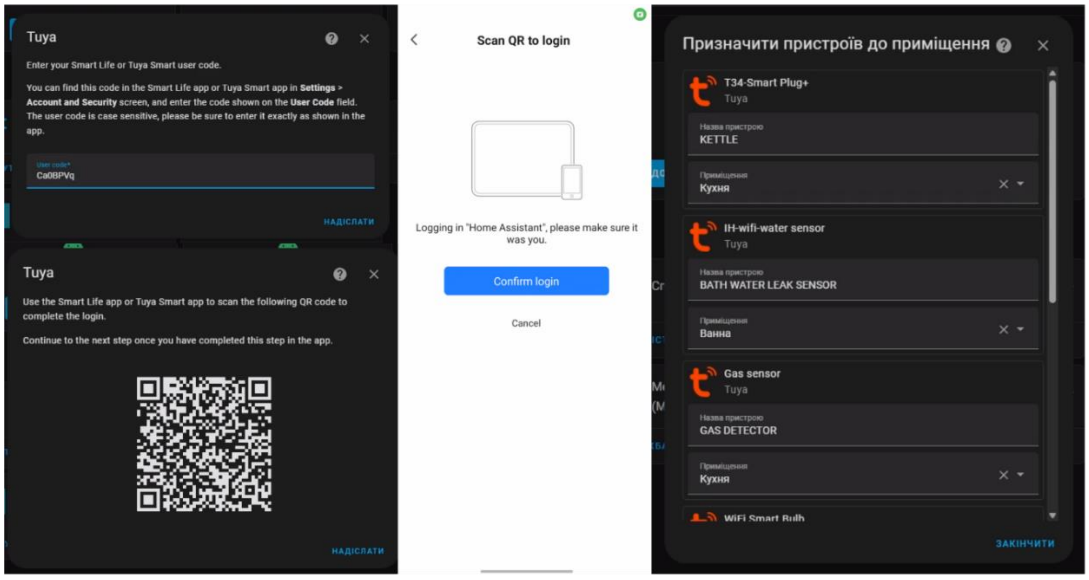

Рисунок 2.6 – Налаштування хмарної інтеграції Tuya для прив’язки IoT пристроїв

Завдяки такій авторизації здійснюється безпосереднє підключення до облікового запису Tuya через API, після чого система отримує доступ до всіх пристроїв, зареєстрованих у відповідному обліковому записі, що дозволяє автоматично імпортувати всю доступну інфраструктуру, включно з освітленням, розумними розетками та іншими IoT-пристроями, які стають відразу видимими у Home Assistant без потреби додаткової локальної конфігурації. Після завершення підключення кожен пристрій автоматично отримує набір відповідних сутностей, які відображають функціональні можливості пристрою: наприклад, у випадку розумної розетки можуть бути сутності для керування вмиканням/вимиканням, а також для моніторингу споживання енергії (див. рис. 2.7).

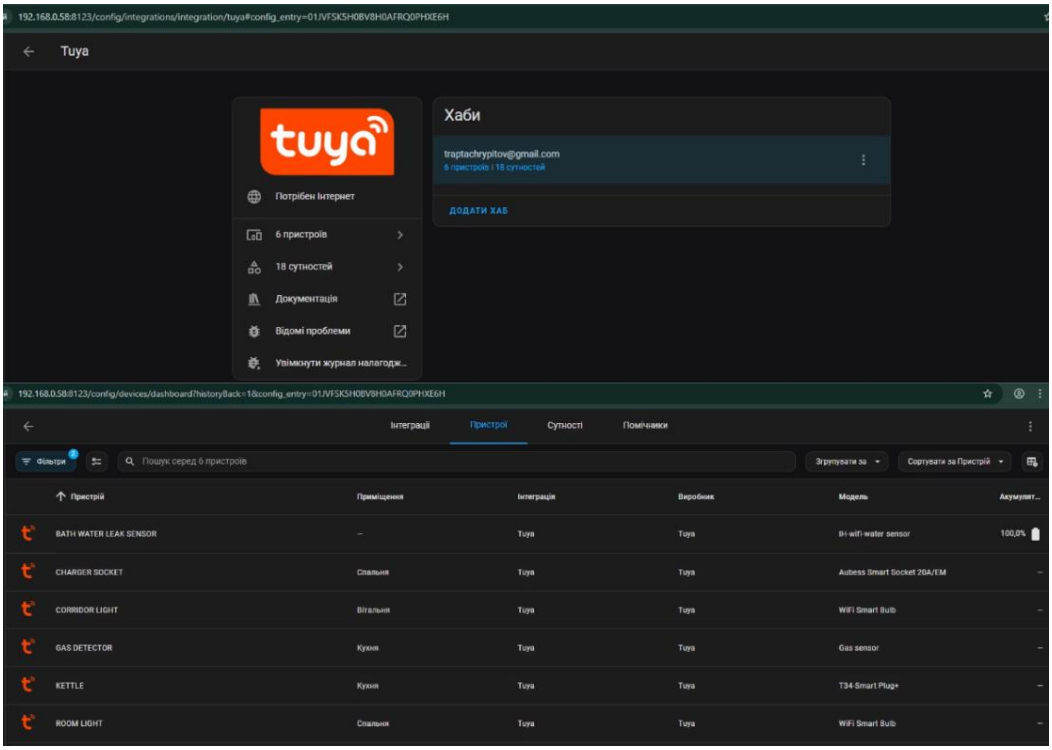

Рисунок 2.7 – Хмарна інтеграція пристроїв Tuya з Home Assistant

Усі ці сутності автоматично додаються до базового дашборду, що створюється при додаванні нових пристроїв, який дозволяє у простій формі керувати всіма підключеними пристроями (див. рис. 2.8) [23, 35-36].

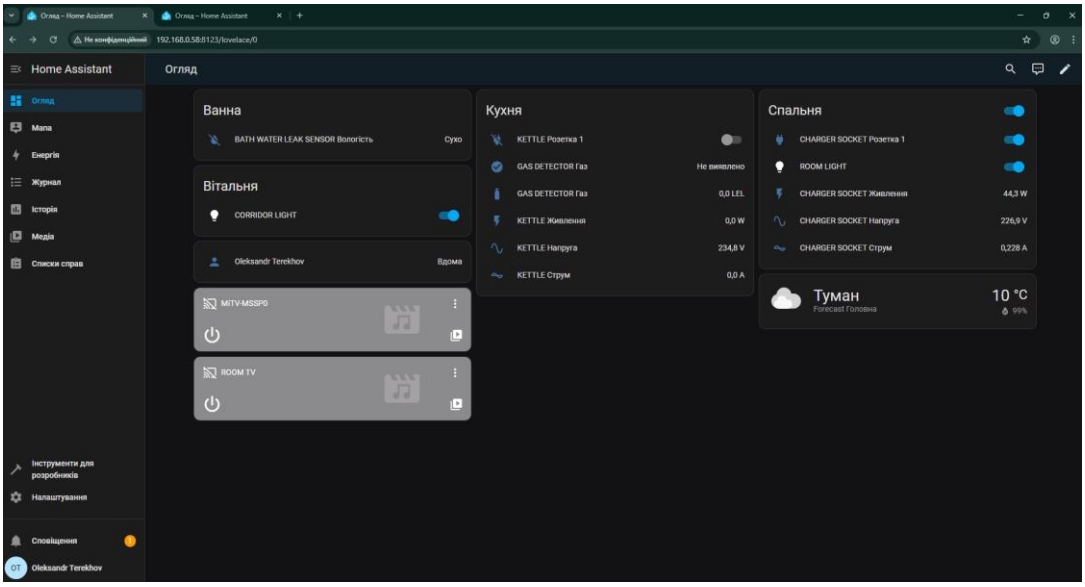

Рисунок 2.8 – Автоматично згенерований дашборд всіх сутностей після хмарної інтеграції Tuya

Такий дашборд фактично є аналогом головного екрана у додатку Smart Life, однак на відміну від мобільної програми його можна гнучко налаштовувати:

— змінювати порядок елементів;

— створювати нові вкладки;

— змінювати зовнішній вигляд елементів керування;

— групувати пристрої за категоріями;

— додавати візуальні блоки (графіки та інші віджети).

Тим не менш, автоматично згенерований дашборд є лише базовим шаблоном, який варто вдосконалювати за допомогою більш функціональних рішень, таких як кастомні картки або готові шаблони оформлення, що дозволяють створювати інтерфейси з розширеною логікою відображення даних.

Незважаючи на зручність та швидкість налаштування, використання інтеграції Tuya у такому хмарному форматі має суттєве обмеження: залежність від стабільного інтернет-з’єднання. У разі втрати зв’язку з хмарою Tuya або проблем з авторизацією API вся система втрачає доступ до пристроїв, що робить її вразливою до зовнішніх чинників, зокрема до збоїв на стороні провайдера або тимчасових неполадок сервісу. Саме ця залежність обмежує можливості автономного управління, а отже постає потреба у впровадженні локальної інтеграції, яка дозволить здійснювати повне управління без залучення сторонніх серверів і забезпечить більшу стабільність, конфіденційність і незалежність системи.

Підключення пристроїв через локальну інтеграцію LocalTuya. Оскільки стандартна інтеграція Tuya в Home Assistant працює через хмару, що одразу стає помітно при першому використанні, то з огляду на загальну мету роботи, яка полягає у повній локалізації керування пристроями, було обрано альтернативне рішення — неофіційне доповнення LocalTuya, доступне лише через Home Assistant Community Store (HACS). Щоб скористатися ним, спершу необхідно встановити сам HACS, що здійснюється досить просто. Потрібна наявність облікового запису GitHub для зв’язку зі стороннім репозиторієм, після чого Home Assistant перезавантажується, і в інтерфейсі з’являється новий пункт HACS, який стає доступним з лівого бокового меню (див. рис. 2.9).

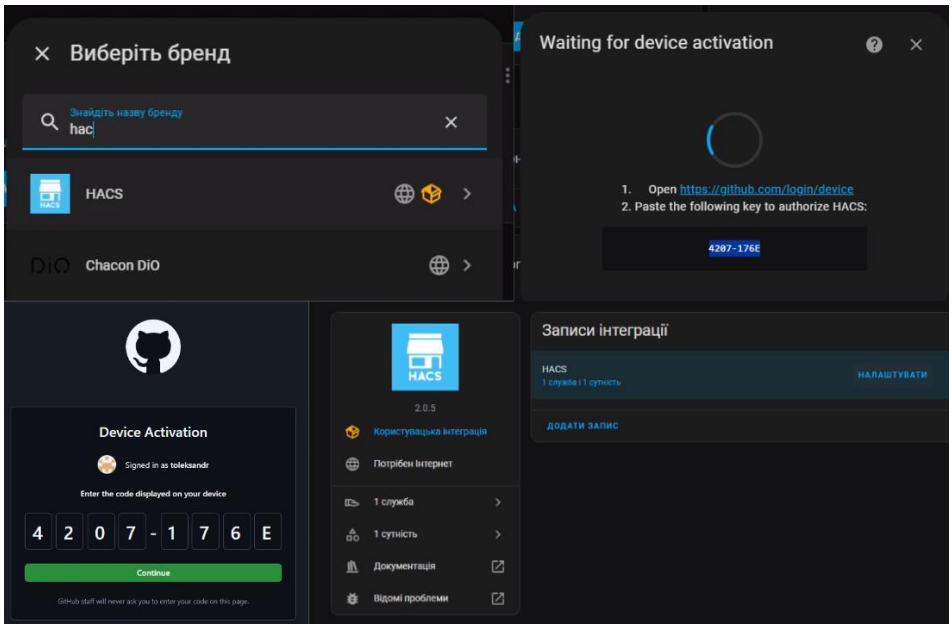

Рисунок 2.9 – Налаштування стороннього магазину HACS для роботи зі користувацькими інтеграціями

Через нього можна завантажити як функціональні інтеграції, так і кастомні візуальні компоненти, зокрема низку стилізованих елементів інтерфейсу для дашбордів, які в подальшому будуть застосовуватись у вигляді карток "mini-card" або "mushroom" (див. рис. 2.10) [24, 35-36].

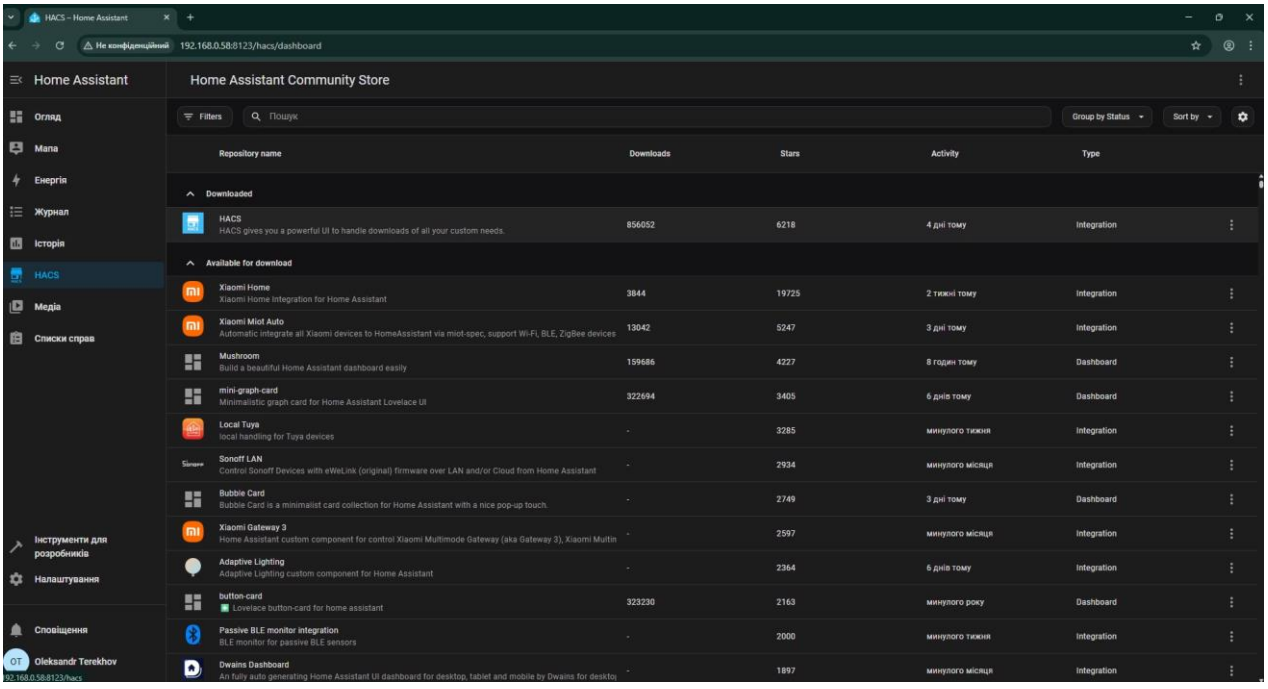

Рисунок 2.10 – Сторінка стороннього магазину HACS

Після встановлення HACS відкривається можливість встановити інтеграцію LocalTuya, яка дозволяє повністю обійти хмарну інфраструктуру Tuya (див. рис. 2.11).

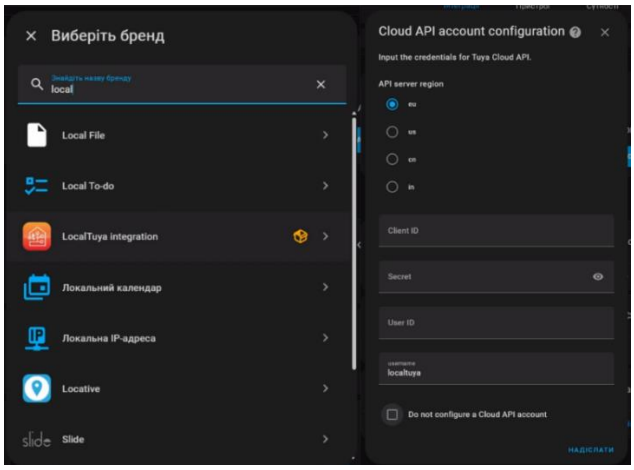

Рисунок 2.11 – Початок інтеграції LocalTuya до Home Assistant

Для повноцінної роботи необхідно також створити акаунт на платформі Tuya IoT Platform — "develop.tuya.com", де здійснюється реєстрація, прийняття всіх умов і створення нового проєкту в розділі "My Cloud Project" (див. рис. 2.12).

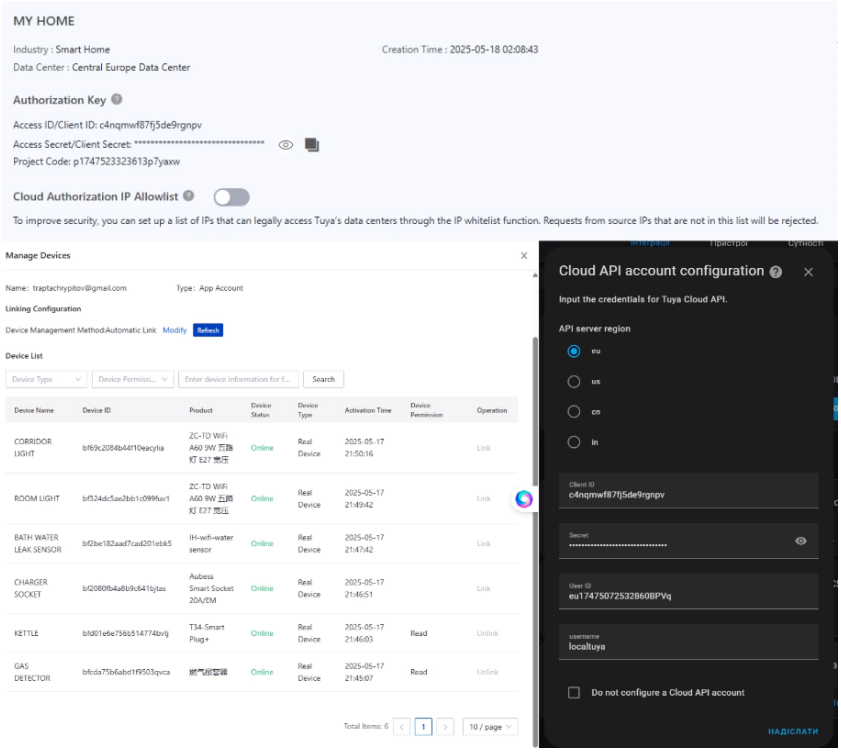

Рисунок 2.12 – Отримання доступу від Tuya IoT Platform для інтеграції LocalTuya

Назва проєкту може бути довільною, наприклад "My Home", оскільки він репрезентує локальну домашню систему. Під час створення проєкту потрібно обрати набір API-сервісів, з яких критично важливим є "Device Status Query", оскільки він дозволяє отримувати дані з пристроїв без звернення до хмари. Після підготовки сторонньої платформи та зв’язку з акаунтом Smart Life потрібно перейти безпосередньо до додавання пристроїв у Home Assistant через інтеграцію LocalTuya. Після встановлення та перезапуску системи в розділі «Інтеграції» з’являється відповідний пункт, де можна розпочати конфігурування нових пристроїв. У меню LocalTuya "Configuration" потрібно обрати "Add a new device", після чого система намагається просканувати локальну мережу. Проте не всі пристрої можуть відобразитись автоматично, отже для частини з них потрібно вручну вводити IP-адресу, локальний ключ (local key) і device ID, які попередньо отримуються через платформу Tuya IoT . Наприклад, для розумних розеток, процес значно простіший. Після натискання "Add a new device" обирається відповідний пристрій, наприклад "Charger Socket" або "Kettle", після чого відкривається інтерфейс з базовими налаштуваннями. У ньому необхідно обрати тип сутностей (entity type), які будуть прив’язані до пристрою, наприклад перемикач або сенсор енергоспоживання. На відміну від хмарної інтеграції, тут немає автоматичного додавання всіх сутностей, тому кожна має налаштовуватись вручну, базуючись на даних із Tuya IoT Platform (див. рис. 2.13)

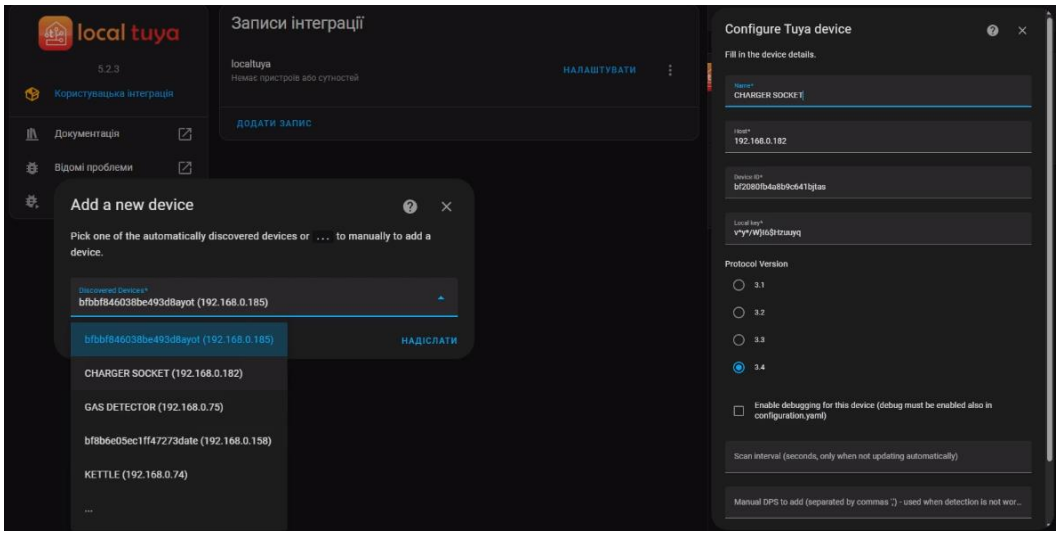

Рисунок 2.13 – Початок ручної конфігурації пристроїв Tuya

Щоб отримати ці дані, у розділі "Device Management" на платформі Tuya потрібно перейти до пункту "Query Device Details", де вводиться "device ID". У результаті формується відповідь з параметрами пристрою, включаючи локальний ключ, який критично важливий для локальної інтеграції. Наступний крок — отримання списку всіх сутностей, які підтримує пристрій, що здійснюється в категорії "Device Control", а саме в запиті "Query Device Properties". Там знову вводиться "device ID", і відповідь містить усі функції та параметри пристрою, які можуть бути налаштовані як окремі сутності в Home Assistant (див. рис. 2.14) [24, 27, 36].

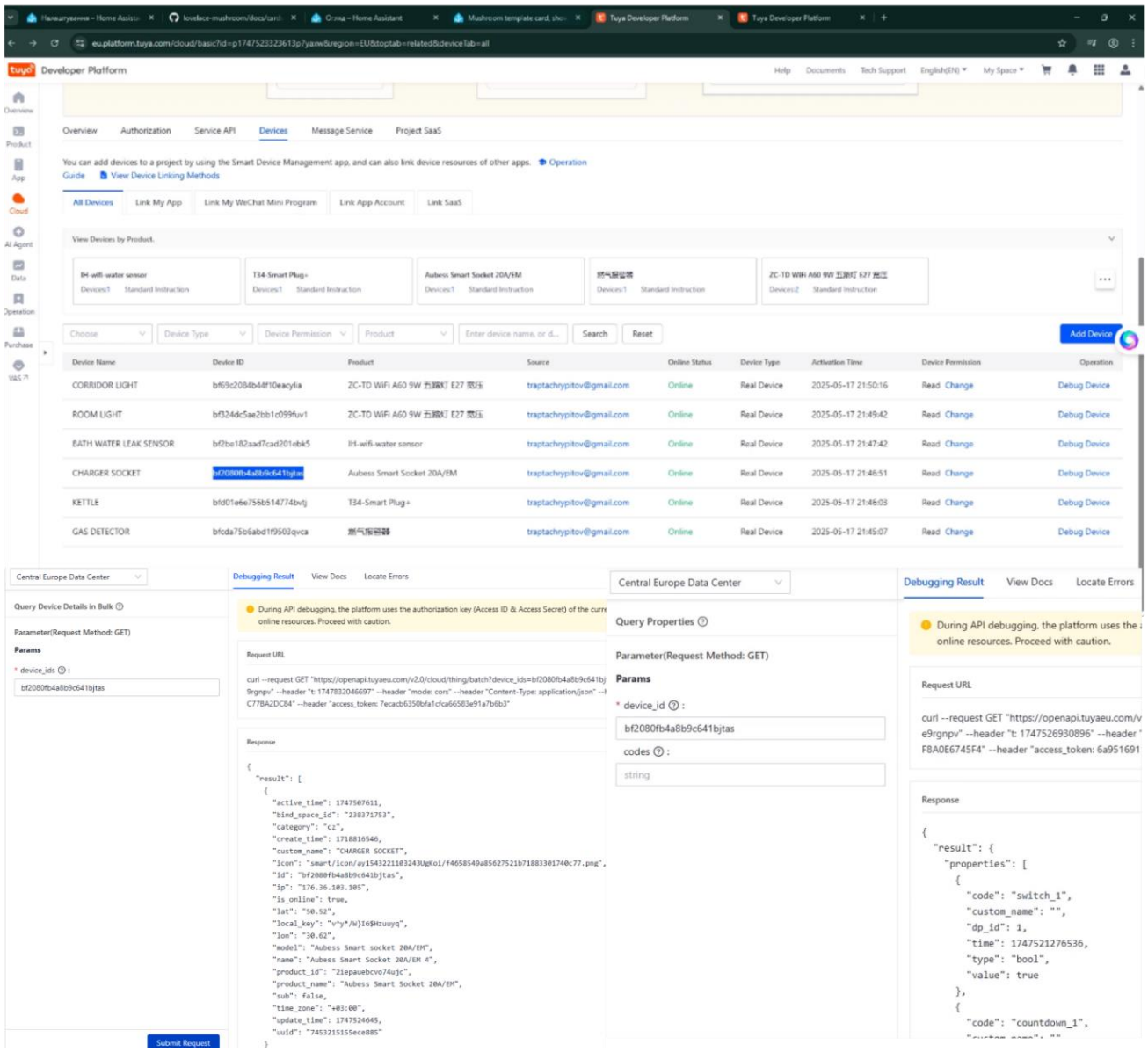

Рисунок 2.14 – Отримання даних про IoT пристрої для локальної прив’язки

Після отримання всіх кодів сутностей (наприклад, для вмикання/вимикання, індикації споживання енергії чи стану мережі) їх поступово додають вручну під час конфігурації кожного пристрою. На прикладі розумних розеток така конфігурація дозволяє додати перемикач живлення, сенсори потужності, напруги, сили струму тощо. Кожна сутність вимагає окремої прив'язки, що ускладнює початкове налаштування, однак дозволяє досягти повного контролю над пристроями без посередництва зовнішніх серверів (див. рис. 2.15).

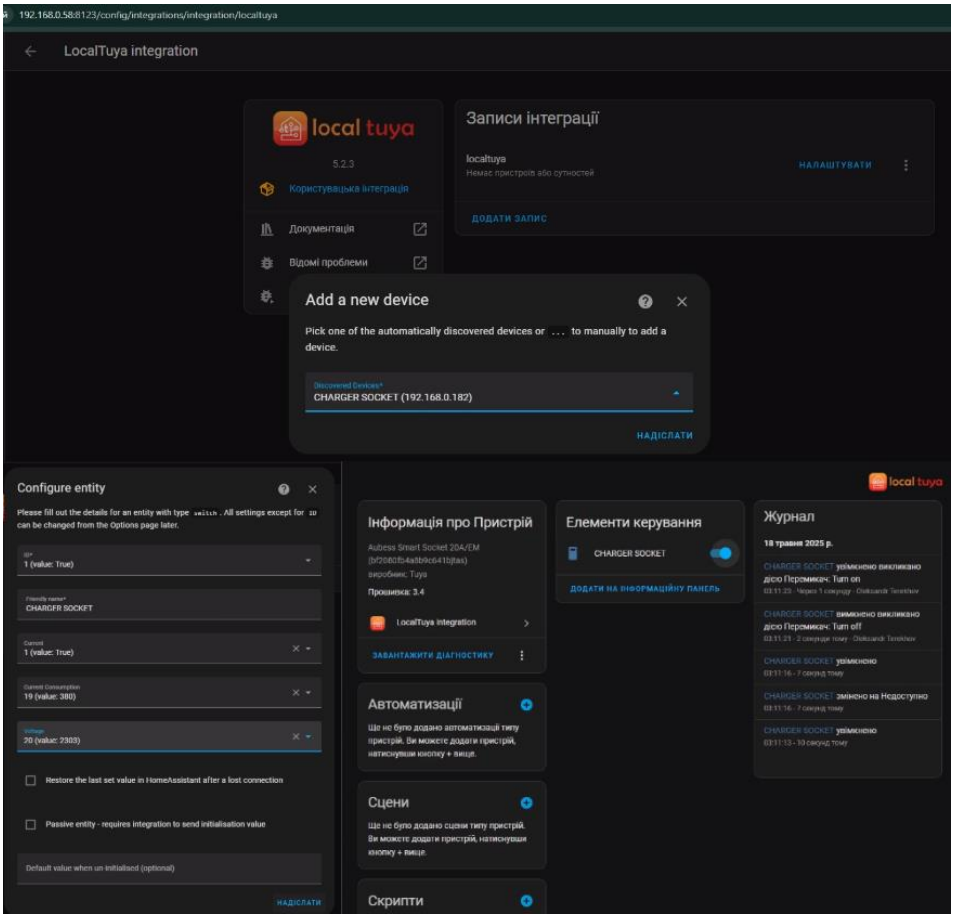

Рисунок 2.15 – Підключення пристроїв через локальну інтеграцію LocalTuya

Після завершення додавання всіх пристроїв — двох розеток, двох ламп, сенсора газу та витоку води — Home Assistant отримує повноцінну локальну систему управління. Всі пристрої функціонують автономно, без потреби в підключенні до хмарної інфраструктури.

Далі система переходить до етапу налаштування візуального представлення даних на дашборді. Завдяки можливостям HACS та карток Mushroom створюється

інформативний, мінімалістичний та зручний для мобільного використання інтерфейс, адаптований для планшетів та смартфонів.

Основна увага зосереджується на створенні інтерфейсу, орієнтованого на автоматизацію, оскільки саме сценарії автоматичної взаємодії пристроїв становлять основу подальшого функціонального розширення. Після реалізації локальної інтеграції та її порівняння зі швидкодією хмарного підходу було підтверджено значну перевагу локального варіанту — управління стало миттєвим, без затримок, які притаманні хмарному рішенню, що доводить ефективність використання LocalTuya як ключового механізму побудови автономної системи розумного дому, де Home Assistant має повний контроль над усіма пристроями без залучення зовнішніх серверів, що підвищує надійність, швидкодію та безпеку загальної архітектури.

Структура взаємодії – Home Assistant та пристрої Tuya через локальну мережу. У контексті взаємодії між Home Assistant та пристроями екосистеми Tuya через локальну мережу структура побудована на принципі прямого з'єднання «точка-точка», де локальний сервер виступає центральним елементом керування та обміну даними з кожним пристроєм у межах локальної інфраструктури. Фізично система базується на віртуалізованому середовищі, де Home Assistant розгорнуто за допомогою VirtualBox шляхом запуску готового дискового образу Home Assistant Operating System. Після старту системи було проведено первинне налаштування, зокрема встановлено стороннє доповнення LocalTuya через HACS, що забезпечує можливість прямого локального зв'язку з пристроями без звернення до хмарних сервісів [25, 35].

Подальша конфігурація полягала в додаванні конкретних пристроїв у вигляді сутностей (entities), кожна з яких відповідала окремому елементу системи — перемикачам, сенсорам, реле або датчикам. Усі пристрої, що використовуються в межах цього середовища, підключені до локального маршрутизатора через Wi-Fi і мають власну IP-адресу, яка або була призначена вручну як статична, або закріплена за допомогою DHCP-резервації, що дозволяє уникати повторного призначення адрес та гарантує стабільність підключення [25, 35].

Передача даних між Home Assistant і пристроями Tuya відбувається через стандартний протокол TCP/IP з використанням порту 6668, що дозволяє безпосереднє керування кожним пристроєм без посередництва хмарного середовища. Через локальну маршрутизацію команда керування надсилається без затримки, а відповідь повертається до локального сервера, що дає змогу реалізувати децентралізовану модель контролю без залежності від інтернету або зовнішніх сервісів [25, 35].

На момент реалізації інтеграції в мережі було під'єднано шість пристроїв: дві розумні розетки, дві розумні лампи, один датчик витоку води та один детектор газу. Кожен з них підтримує локальну роботу через LocalTuya і автоматично розпізнається у Home Assistant як окрема сутність. Таким чином, локальний сервер забезпечує пряме двостороннє управління всіма пристроями через внутрішню інфраструктуру, де взаємодія відбувається безпосередньо, минаючи зовнішні шлюзи чи централізовані обчислювальні ресурси, що підвищує як швидкодію системи, так і її автономність.

Використання YAML-конфігурацій і Lovelace Dashboard. У системі Home Assistant конфігурація зберігається у форматі YAML, що розшифровується як "Yet Another Markup Language" і фактично є текстовим форматом, орієнтованим на зручне сприйняття людиною. Через YAML-файли визначаються ключові параметри інтеграції пристроїв, зовнішнього вигляду інтерфейсу, сценаріїв автоматизації, а також налаштування служб та додатків. Хоча багато з цих елементів можна редагувати у візуальному редакторі самого Home Assistant, саме робота з YAML забезпечує більшу гнучкість, контроль та точність у налаштуваннях, особливо коли йдеться про складні ланцюги автоматизації або тонке оформлення інтерфейсу [26].

Наприклад, сценарій автоматизації вимкнення розетки у разі витоку води описується простою YAML-конструкцією, де задається тригер спрацювання сенсора, який активує відповідну дію — вимкнення певного перемикача. Така структура дозволяє точно вказати умови, за яких відбувається певна дія, і при цьому уникнути залежності від сторонніх інструментів чи шаблонів. Подібна

автоматизація реалізується локально, реагує миттєво і є прозорою у налаштуванні, що значно полегшує налагодження роботи домашньої мережі керованих пристроїв.

Паралельно з конфігураційним файлом YAML у Home Assistant використовується система Lovelace Dashboard — гнучкий інтерфейс візуалізації, що дозволяє створювати персоналізовані панелі керування з розширеною функціональністю. У межах однієї панелі можливо відображати поточний стан пристроїв, таких як розетки, сенсори або лампи, а також додавати елементи керування:

— кнопки;
— перемикачі;
— графіки історії;
— сповіщення про спрацювання;
— таймери;
— сцени;
— інші інтерактивні блоки.

Панелі можуть формуватися вручну через YAML або у вбудованому редакторі безпосередньо через веб-інтерфейс (див. рис. 2.16) [26].

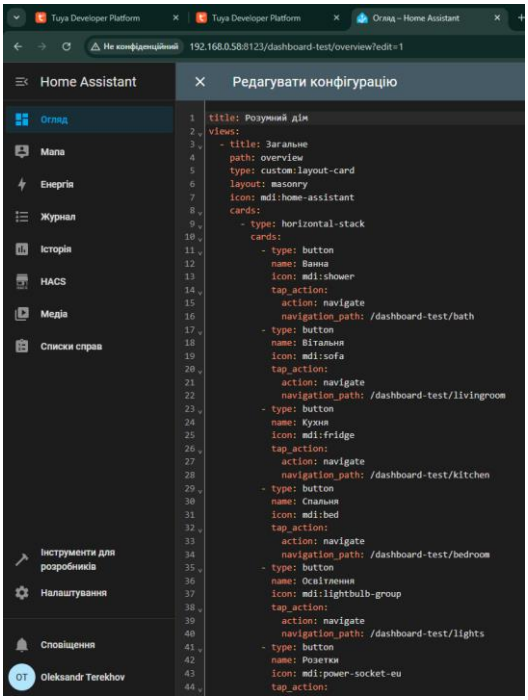

Рисунок 2.16 – Приклад YAML-конфігурації Lovelace Dashboard

Приклад такої панелі може виглядати як перелік сутностей у форматі "type: entities", де відображається група пристроїв однієї кімнати — наприклад, кухні — з назвами, які користувач задає самостійно. У такому випадку інтерфейс дашборду стає зрозумілим, компактним і легким у використанні, особливо на мобільних пристроях. Завдяки адаптивному дизайну він однаково добре працює як на смартфоні, так і на планшеті, зберігаючи при цьому повну функціональність доступу до пристроїв локальної мережі (див. рис. 2.17-2.18).

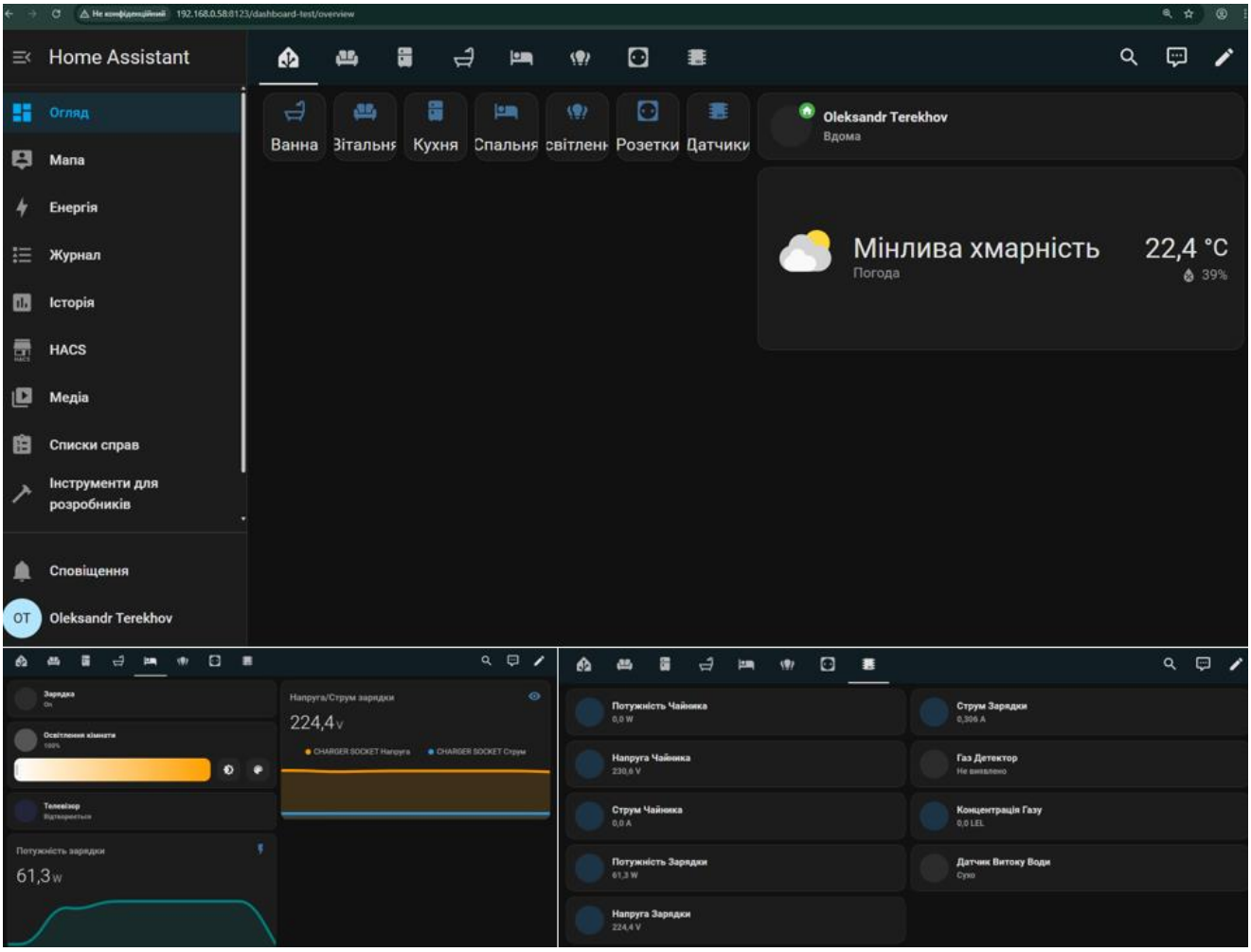

Рисунок 2.17 – Власний дашборд всіх сутностей після локальної інтеграції LocalTuya (веб-застосунок)

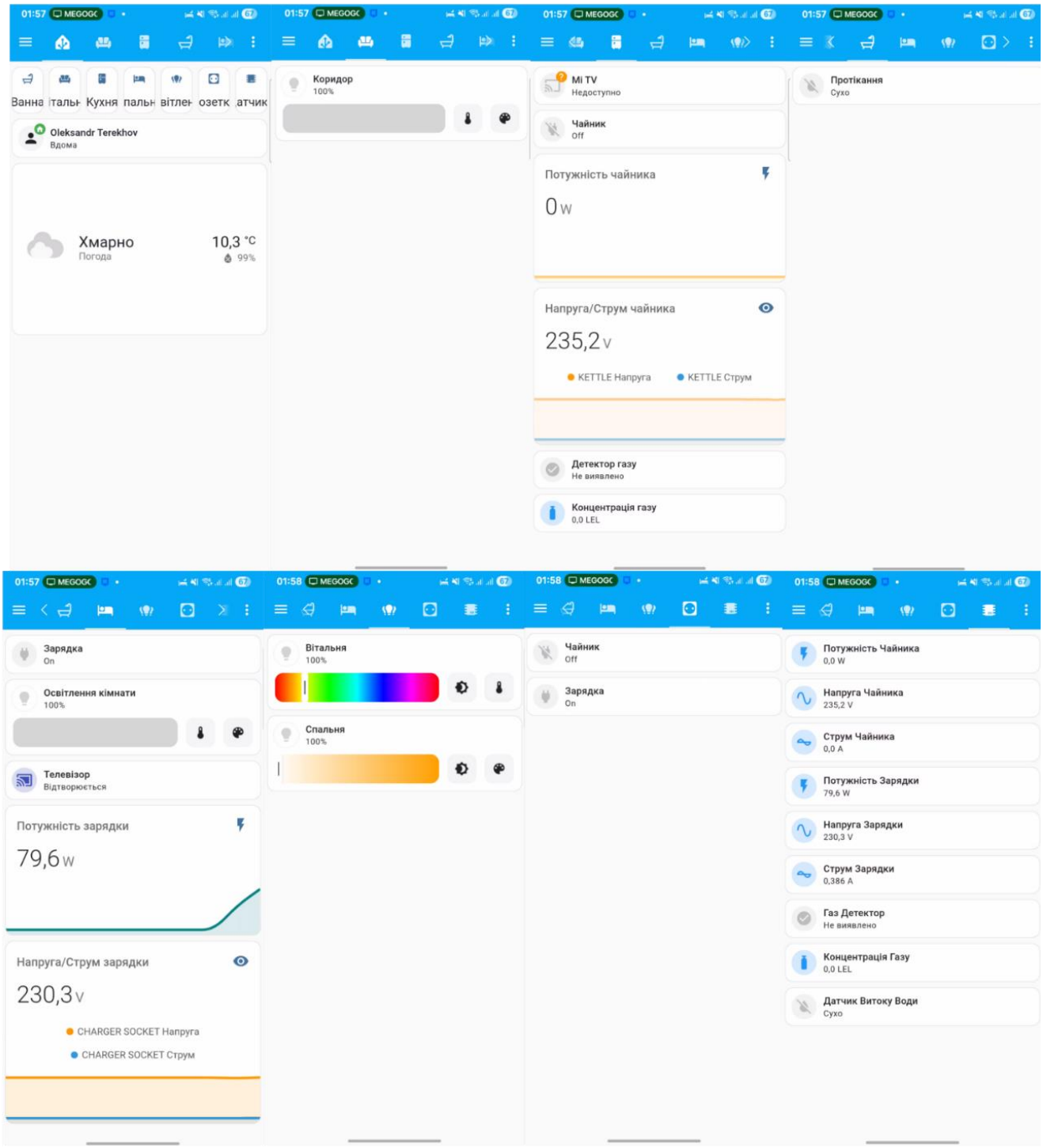

Рисунок 2.18 – Власний дашборд всіх сутностей після локальної інтеграції LocalTuya (мобільний застосунок)

Подальше вдосконалення інтерфейсу здійснюється через використання кастомних карт, які надаються спільнотою в репозиторії HACS:

— Mini Graph Card;

— Mushroom.

Саме ці розширення використовувалися для створення власного кастомного дашборду, в основі якого лежать компоненти Lovelace, адаптовані під потреби мобільного користування. Отриманий інтерфейс не лише зручний і зрозумілий, але й візуально привабливий, поєднуючи функціональність з дизайном. Незважаючи на те, що більшість налаштувань здійснювалося через візуальний редактор, у процесі налаштування все ж активно використовувалися фрагменти YAML-конфігурацій, що дозволило гнучко поєднати обидва підходи — ручний і візуальний — для досягнення бажаного результату у вигляді першого тестового варіанту інтерфейсу керування, адаптованого під реальні потреби локального середовища.

2.3 Реалізація логіки управління та автоматизацій

Однією з основних переваг відкритих систем керування на зразок Home Assistant є можливість побудови гнучкої, локалізованої та орієнтованої на конкретні умови логіки автоматизації, яка значно перевершує обмеження типової хмарної інфраструктури, наприклад, у додатку Smart Life. Завдяки локальному сценарному підходу автоматизація може базуватись не лише на простих подіях або часу, а й на поєднанні умов, станів, контексту поведінки користувача, пріоритетів середовища та інших змінних, що дозволяє побудувати складні, адаптивні схеми керування без необхідності підключення до зовнішніх серверів. Такий підхід особливо ефективний у ситуаціях, коли потрібно зменшити затримки, підвищити стабільність виконання сценаріїв та забезпечити незалежність від нестабільності інтернет-з’єднання [24-27].

Побудова сценаріїв – оповіщення про виявлення газу та води. Одними з базових, але критично важливих сценаріїв автоматизації в системі Home Assistant є оповіщення про виявлення газу та витік води. Раніше подібна логіка була передбачена у хмарному середовищі Tuya (платформа Smart Life), де типові дії

активувалися автоматично при додаванні відповідних сенсорів. Проте при переході на локальну інфраструктуру з використанням інтеграції LocalTuya, усі подібні сценарії потрібно створювати вручну, адже без цього сенсори лише фіксуватимуть зміну стану без жодної реакції системи.

У середовищі Home Assistant реалізація сценарію виявлення газу базується на відстеженні зміни стану сутності "binary_sensor.gas_detector_gas ". При переході сенсора у стан on (тобто при виявленні газу), спрацьовує тригер, що ініціює надсилання push-сповіщення через вбудований мобільний застосунок Home Assistant. Повідомлення надсилається з високим пріоритетом і мінімальним часом затримки, завдяки сервісу "notify.mobile_app_sm_s928b", що дозволяє оперативно поінформувати користувача про небезпеку (див. рис. 2.19).

Аналогічний підхід використовується для сценарію виявлення витоку води у ванній кімнаті. У цьому випадку спрацьовує сенсор вологості "binary_sensor.bath_water_leak_sensor_moisture", який активує ту ж саму дію — push-сповіщення на смартфон. Повідомлення містить чіткий заголовок і текст, що вказує на характер та місце інциденту, закликаючи до негайної перевірки. Цей сценарій також потребує ручного створення при локальному керуванні, оскільки інтеграція LocalTuya не містить вбудованих шаблонів реагування на події (див. рис. 2.19).

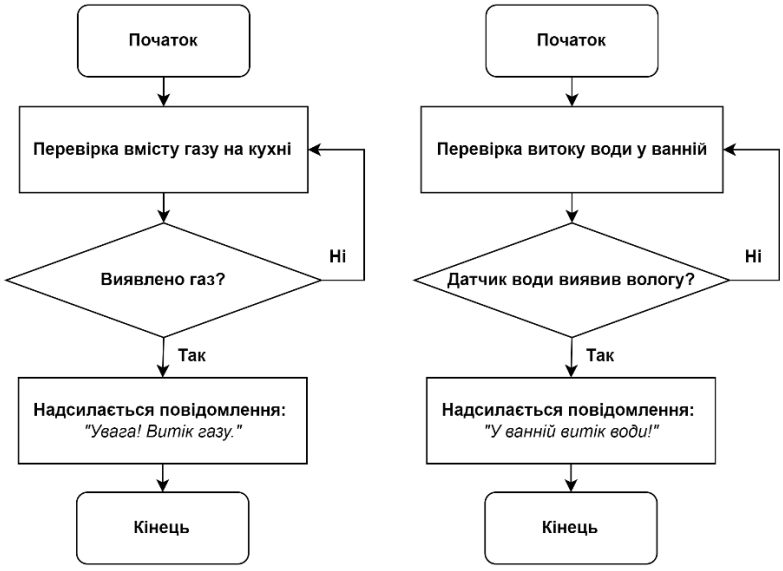

Рисунок 2.19 – Алгоритми автоматизації оповіщення про витік газу та води

Таким чином, ефективність системи залежить не лише від наявності датчиків, але й від правильно побудованої логіки реагування. Автоматизація на локальному рівні забезпечує не лише автономність від хмари, а й вищу швидкість реагування та більшу надійність у критичних ситуаціях.

Побудова сценаріїв – вимикання зарядного пристрою при низькому навантаженні. Ще одним прикладом реалізації сценарію є автоматичне вимкнення зарядного пристрою при низькому навантаженні, яке певною мірою вже було доступне на попередній платформі Smart Life, де параметри контролю енергоспоживання розумної розетки задавалися вбудовано, без потреби створювати окремі автоматизації. У хмарній інфраструктурі Tuya подібний функціонал виступав у вигляді одного з параметрів налаштування пристрою, наприклад, у випадку розетки типу "CHARGER SOCKET", де вбудовані опції дозволяли визначити поріг потужності, після досягнення якого пристрій автоматично вимикався. Однак такий підхід був обмежений фіксованою логікою та не дозволяв гнучко змінювати умови чи поєднувати їх з іншими подіями (див. рис. 2.20).

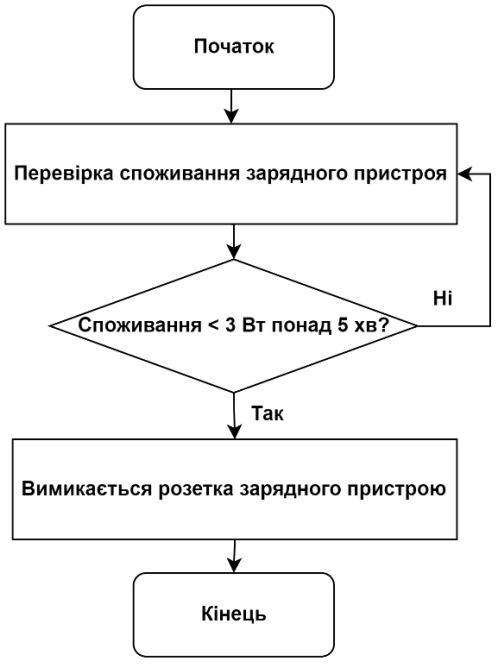

Рисунок 2.20 – Алгоритм автоматизації вимкнення зарядного пристрою при низькому навантаженні

У середовищі Home Assistant подібний сценарій реалізовано через повністю кастомізовану автоматизацію, яка реагує на зниження навантаження нижче за 3 Вт протягом понад п’яти хвилин і вимикає відповідну розетку, що дає змогу точно контролювати процес заряджання пристроїв і уникати неефективного енергоспоживання. YAML-сценарій передбачає використання сенсора потужності "sensor.charger_socket_socket_1_power", який, у разі коли його значення опускається нижче за 3 Вт і це триває не менше п’яти хвилин, активує команду "switch.turn_off" для відповідної розетки "switch.charger_socket_socket_1". Таким чином, сценарій діє повністю локально без хмарної залежності та забезпечує додаткову економію електроенергії та підвищення безпеки.

Побудова сценаріїв – автоматизація увімкнення та вимкнення чайника вранці. Наступний приклад сценарію демонструє інший рівень гнучкості та функціональності, який складно або неможливо реалізувати у середовищі Smart Life чи на базі хмарної Tuya через обмеження логіки та налаштувань. У випадку автоматичного вмикання та вимикання чайника вранці реалізовано комбінований сценарій, який враховує як час, так і присутність користувача вдома. Раніше подібне можна було частково зробити шляхом фіксації часу вмикання розетки у визначені дні, проте перевірку на присутність реалізувати не вдавалось, оскільки такі умови не підтримувались на рівні хмарного середовища.

У Home Assistant це реалізовано завдяки використанню двох тригерів за часом — о 06:30 та 06:35 — із зазначенням днів тижня (понеділок, середа, п’ятниця), а також перевіркою стану присутності користувача (наприклад, "person.toleksandr"). Якщо обидві умови виконано, сценарій активується, і за допомогою конструкції "choose" у YAML відбувається або вмикання, або вимикання розетки "switch.kettle_socket_1", залежно від того, який саме тригер спрацював (див. рис. 2.21).

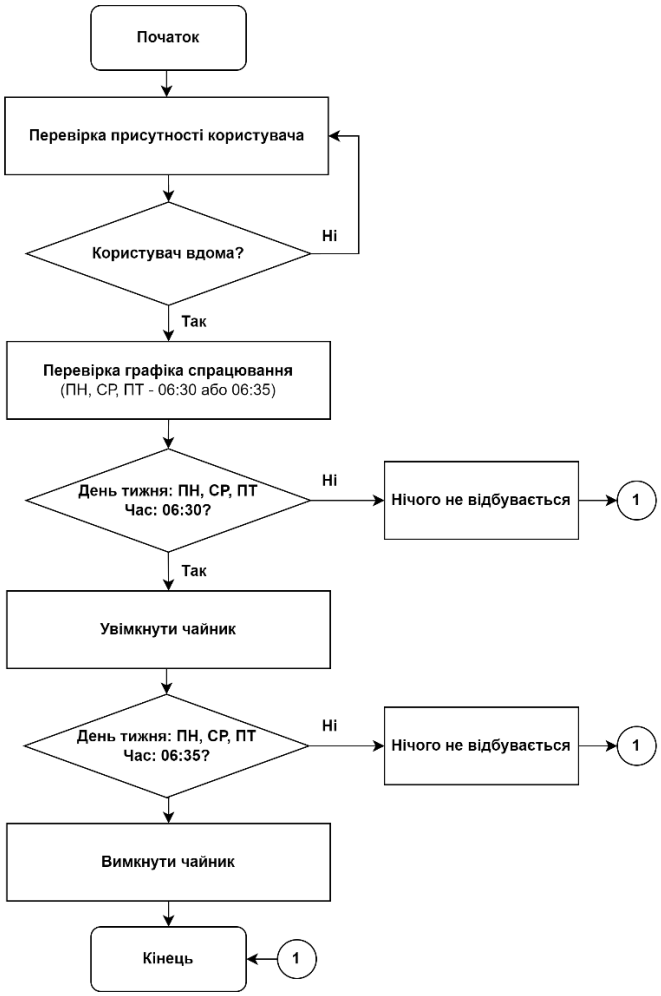

Рисунок 2.21 – Алгоритм автоматизації вмикання чайника вранці

Такий підхід дозволяє уникати дублювання автоматизацій і створювати більш гнучкі сценарії з однією конфігурацією, що не тільки підвищує зручність, але й спрощує керування розумними пристроями в межах локальної інфраструктури.

Побудова сценаріїв – автоматизація світла ввечері та вночі. Автоматизація світла у вечірній та нічний час демонструє відмінності в підході до створення сценаріїв у порівнянні з платформою Smart Life, яка базується на хмарній інфраструктурі Tuya. У межах цієї реалізації застосовується гнучкіша логіка побудови, яка дозволяє поєднати декілька умов і подій у межах одного сценарію за допомогою конструкції "choose". У конкретному випадку використовується автоматичне ввімкнення освітлення в коридорі та спальні одразу після заходу сонця, орієнтуючись на локальні астрономічні дані, що динамічно змінюються протягом року, наприклад, у травні 2025 року захід сонця в Києві настає приблизно

о 20:43. Освітлення вмикається безпосередньо після цієї події, якщо тригером стає "sunset", тобто захід сонця, і стосується одразу двох джерел світла — у вітальні та спальні.

Друга частина автоматизації, реалізована у тому ж самому сценарії, відповідає за вимкнення освітлення в обох приміщеннях о 00:30, що, наприклад, може співпадати з часом, коли користувачі завершують активність і готуються до сну. Такий підхід дозволяє уникнути дублювання кількох окремих автоматизацій і централізовано керувати як увімкненням, так і вимкненням світла в межах єдиного логічного блоку. Завдяки блоку "choose" система спершу ідентифікує, яка саме подія спрацювала — захід сонця або вказаний час уночі — і лише після цього виконує відповідну дію. Умови при цьому можуть бути залишені порожніми, якщо освітлення потрібно вмикати й вимикати незалежно від наявності інших контекстів, наприклад, присутності людини вдома або дня тижня (див. рис. 2.22).

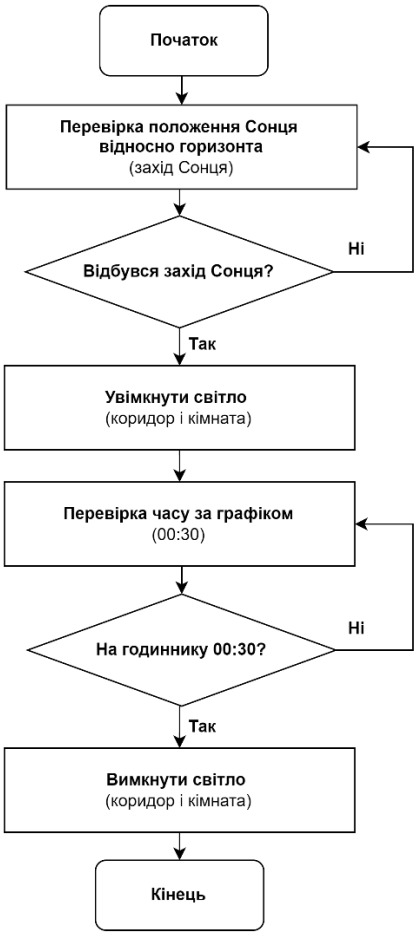

Рисунок 2.22 – Алгоритм автоматизації освітлення у вечірній та нічний час

Таке поєднання логіки на основі астрономічних подій та фіксованих моментів часу створює універсальний механізм освітлення для вечірнього та нічного часу, який не потребує постійного втручання або ручного керування. Уся структура автоматизації впроваджена таким чином, щоб уникнути конфліктів або одночасного виконання кількох дій, тому встановлено режим "single", що унеможливлює повторний запуск сценарію, якщо попередній ще виконується. Цей підхід демонструє ефективність і масштабованість реалізації автоматизованих рішень на базі Home Assistant з використанням логіки подій, характерної для фізичних умов середовища, таких як захід сонця, та сталих звичок користувачів, як-от фіксований час вимкнення світла вночі.

Побудова сценаріїв – автокерування при вході/виході з дому з перевіркою часу. Автоматизація автокерування при вході та виході з дому з перевіркою часу охоплює широку логіку взаємодії пристроїв на основі поточного розташування користувача й часу доби, що дозволяє досягти максимальної адаптивності керування без потреби постійного втручання. У цьому випадку реалізовано сценарій, який відрізняється від простіших варіантів, що передбачають лише фіксований час вмикання або вимикання пристроїв. Наприклад, розетка для чайника, яка може вмикатись вручну або залишатися увімкненою через недосконалий попередній сценарій, у новій логіці обробляється автоматично при поверненні користувача додому, незалежно від того, коли саме це відбувається, що усуває залежність від попередньо заданого часу та враховує реальні обставини.

Повернення додому може бути непередбачуваним: затори, зміни в розкладі чи інші затримки часто не дозволяють заздалегідь визначити точний час входу до будинку. У зв’язку з цим розетки вмикаються одразу після того, як система зафіксувала зміну стану присутності користувача. Однак світло активується лише за умови, що на момент повернення вже настав захід сонця, що дозволяє уникнути зайвого енергоспоживання вдень і при цьому гарантує комфорт в умовах сутінків або темряви. Якщо ж користувач повертається ще до заходу, світло не вмикається, але розетки, такі як для чайника чи зарядного пристрою, вже активні, що забезпечує готовність до використання побутових приладів одразу після входу в дім.

Вихід із дому також обробляється розумно: замість негайного вимкнення всіх пристроїв застосовується п’ятихвилинна затримка, яка дозволяє системі оцінити, чи не є вихід тимчасовим. Наприклад, якщо користувач викидає сміття або вийшов ненадовго, сценарій не запускається, доки не сплине цей буфер часу. Якщо протягом затримки не відбулося повернення, всі розетки й світло вимикаються автоматично, що дозволяє уникнути марного споживання електроенергії без ризику відключити щось важливе у момент, коли ще присутність користувача не була остаточно перервана. Такий підхід забезпечує не лише зручність, а й безпеку, виключаючи випадки залишення ввімкненого обладнання вдома (див. рис. 2.23).

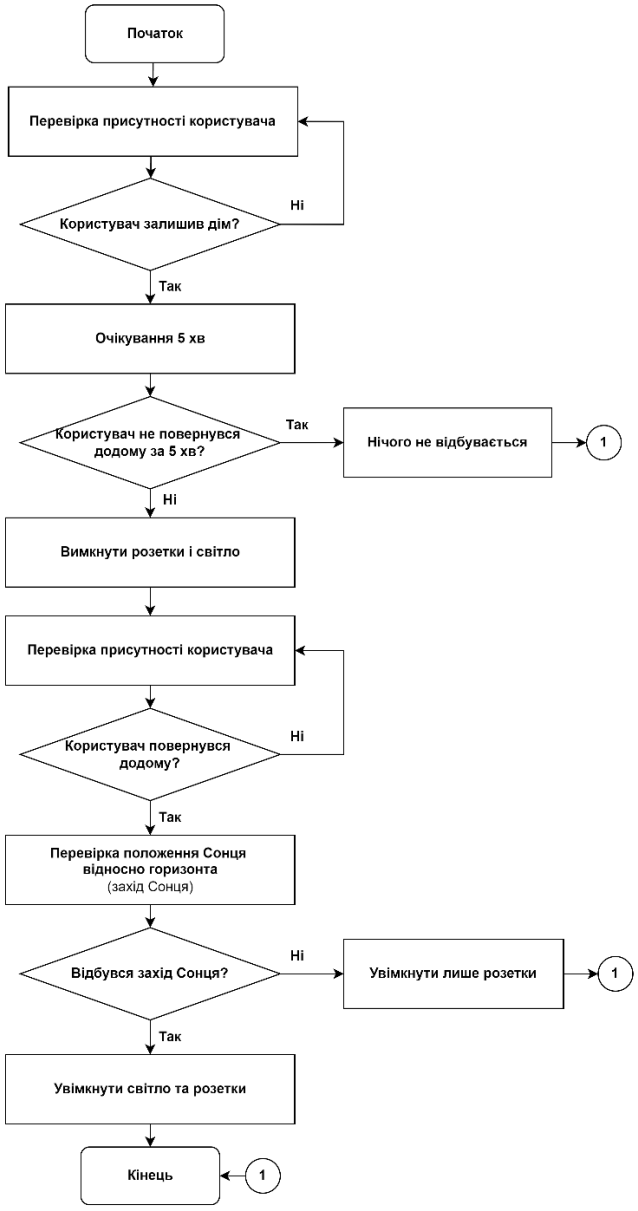

Рисунок 2.23 – Алгоритм автоматизації керування при вході/виході з будинку

Загалом реалізований сценарій демонструє набагато глибший рівень логіки та адаптації порівняно з можливостями платформи Tuya, де подібні рішення можна реалізувати лише через множинні окремі автоматизації, які при цьому не завжди працюють узгоджено. У Home Assistant же структура сценаріїв дозволяє централізовано поєднувати кілька дій у межах однієї автоматизації з розгалуженою логікою залежно від тригерів, а отже, усі елементи працюють синхронно й цілісно. Завдяки наявності чітко визначених умов, можливості застосування перевірок на рівні часу заходу сонця, затримок і поточного стану користувача, система не просто вмикає або вимикає пристрої, а адаптується до реального сценарію поведінки мешканця будинку, реагуючи на конкретні дії [26].

Ключовим моментом є те, що вся логіка виконується локально, тобто без залучення зовнішніх хмарних серверів, як у випадку з Tuya, що дозволяє уникнути затримок у виконанні дій і значно підвищує надійність — кожна дія спрацьовує миттєво після виникнення тригера. Тоді як у хмарних системах навіть кількасекундна затримка або її повна відсутність у відповідь на зміну стану можуть призвести до невдалого виконання сценарію, у випадку з Home Assistant усе працює негайно й без збоїв. Структура на базі "choose" дає змогу чітко розділяти дії на дві логічні гілки — для повернення додому та для виходу з нього — кожна з яких реалізує послідовність дій з урахуванням часу та стану користувача. Така побудова створює стабільне, розширюване й інтелектуальне середовище управління пристроями в межах локальної автоматизованої системи [26].

2.4 Тестування системи в умовах автономної роботи

Після реалізації локальної інтеграції пристроїв Tuya у середовищі Home Assistant за допомогою стороннього доповнення LocalTuya виникла необхідність перевірки ефективності та стабільності такої системи в умовах повної автономності, зокрема без наявності підключення до глобальної мережі Інтернет.

Подібне тестування мало на меті встановити, наскільки система здатна забезпечити:

— безперебійну роботу;

— стабільність автоматизацій;

— мінімальні затримки у виконанні команд;

— високий рівень безпеки;

— повний контроль над керованими об’єктами.

Робота пристроїв у локальній мережі без Інтернету. Оскільки за замовчуванням екосистема Tuya передбачає централізовану хмарну модель керування з обов’язковим з’єднанням з інтернетом, ключовим етапом випробування стала симуляція повної автономності шляхом відключення маршрутизатора від глобальної мережі. Пристрої, що були інтегровані саме через LocalTuya, залишилися повністю функціональними, автоматизації виконувались у штатному режимі, взаємодія між об’єктами продовжувалась без збоїв і затримок, а сам сервер Home Assistant, розгорнутий на локальному обладнанні, не зазнавав жодних проблем у зв’язку з відсутністю доступу до Інтернету. Робота системи у межах лише локальної мережі підтвердила, що її функціонування не залежить від зовнішніх API, серверів чи хмарних механізмів, а контроль і логіка залишаються повністю на стороні локального вузла (див. рис. 2.24).

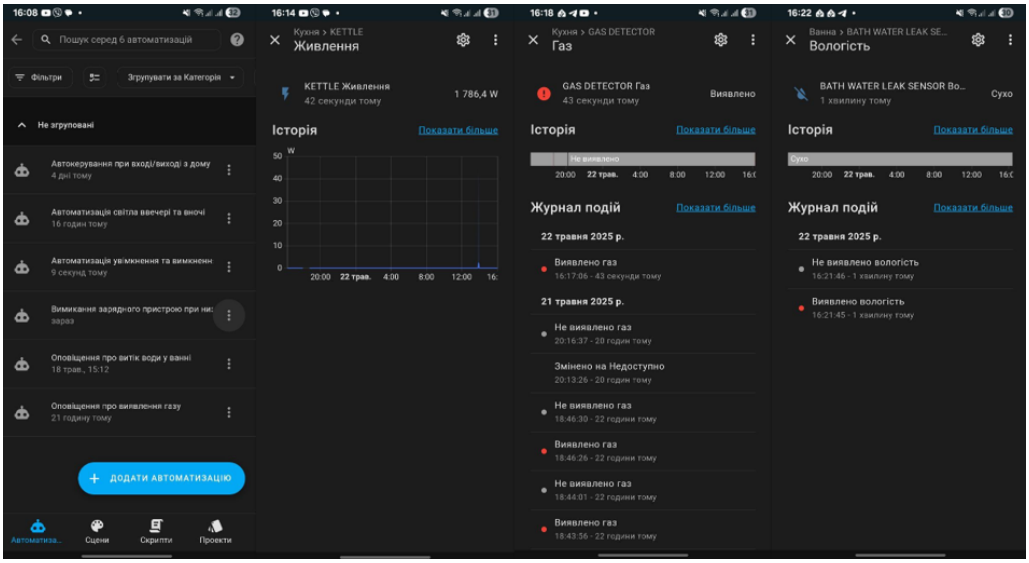

Рисунок 2.24 – Робота пристроїв без підключення до Інтернету

Автоматизації, створені в Home Assistant, зокрема вимкнення розеток після 5-хвилинної бездіяльності, ввімкнення світла після заходу сонця або вимкнення освітлення у певний нічний час, продовжували виконуватись так, як і раніше, демонструючи, що навіть без зв'язку з хмарою зберігається повна логіка обробки тригерів. Крім того, інтерфейс Home Assistant у веб-браузері залишився доступним у локальній мережі, забезпечуючи повноцінний контроль над пристроями, як і офіційний мобільний додаток Home Assistant, який також працював виключно локально. На відміну від цього, застосунок Smart Life, який раніше слугував основним інструментом взаємодії з Tuya-пристроями, втратив здатність керування, адже повністю залежить від хмарної інфраструктури і без Інтернету перетворюється на нефункціональний інтерфейс.

Керування, таким чином, можливо лише з тієї ж локальної мережі, в якій знаходиться сервер Home Assistant. За відсутності тунелювання через VPN або прямого портування ззовні керування неможливе, що з одного боку є обмеженням, а з іншого — підсилює безпеку і незалежність від сторонніх сервісів. Такий режим функціонування особливо актуальний у критичних сценаріях — виявлення витоків, запобігання пожежам, аваріям, автоматичне освітлення, знеструмлення потенційно небезпечних приладів та забезпечення загальної побутової стабільності, де затримка навіть у кілька секунд може мати значення.

Вимірювання затримок при виконанні сценаріїв. Для об'єктивної оцінки швидкодії виконання автоматизацій у різних умовах, зокрема з підключенням до хмарного сервісу Tuya IoT Cloud та у локальному середовищі з інтеграцією через LocalTuya у Home Assistant, була проведена серія вимірювань затримок між ключовими подіями. Вимірювання здійснювалися за допомогою точного фіксування часу між сигналом спрацювання датчика (наприклад, виявлення витоку води) та відповідною дією (надсилання сповіщення). Усі тести проводилися в однакових умовах, що дозволило виявити реальні відмінності у продуктивності обох підходів: спершу вимірювання здійснювалися в умовах централізованої моделі Tuya IoT Cloud із доступом до Інтернету через застосунок Smart Life, а потім

повторювалися в локальній мережі через Home Assistant з інтеграцією LocalTuya та повним відключенням Інтернету.

Таблиця 2.3 – Вимірювання затримок при виконанні імітації сценаріїв

Сценарій	**Затримка через Tuya IoT Cloud**	**Затримка через LocalTuya**
Виявлення витоку → отримання повідомлення	3.2–4.5 сек	0.3–0.5 сек
Споживання менше 3W → деактивація розетки	2.1–3.0 сек	0.2–0.4 сек
Натискання кнопки на інтерфейсі	1.5–2.5 сек	0.1–0.2 сек
Спрацювання таймеру	1.8–2.8 сек	0.1–0.3 сек

Результати тестів (див. табл. 2.3) наочно демонструють суттєву різницю у швидкодії. Наприклад, сценарій виявлення витоку та надсилання повідомлення в хмарному варіанті виконується із затримкою в межах 3,2–4,5 секунди, у той час як у локальному середовищі ця затримка не перевищує 0,5 секунди. Інші сценарії, як-от деактивація розетки, натискання кнопки в інтерфейсі чи спрацювання таймера, також демонструють скорочення затримки у 8–10 разів. Така різниця пояснюється відсутністю потреби у передачі даних на віддалений сервер: інформація не проходить через глобальні маршрути, не потребує зовнішньої обробки, а обмін здійснюється виключно в межах локальної мережі, що значно зменшує час реакції системи (див. рис. 2.25).

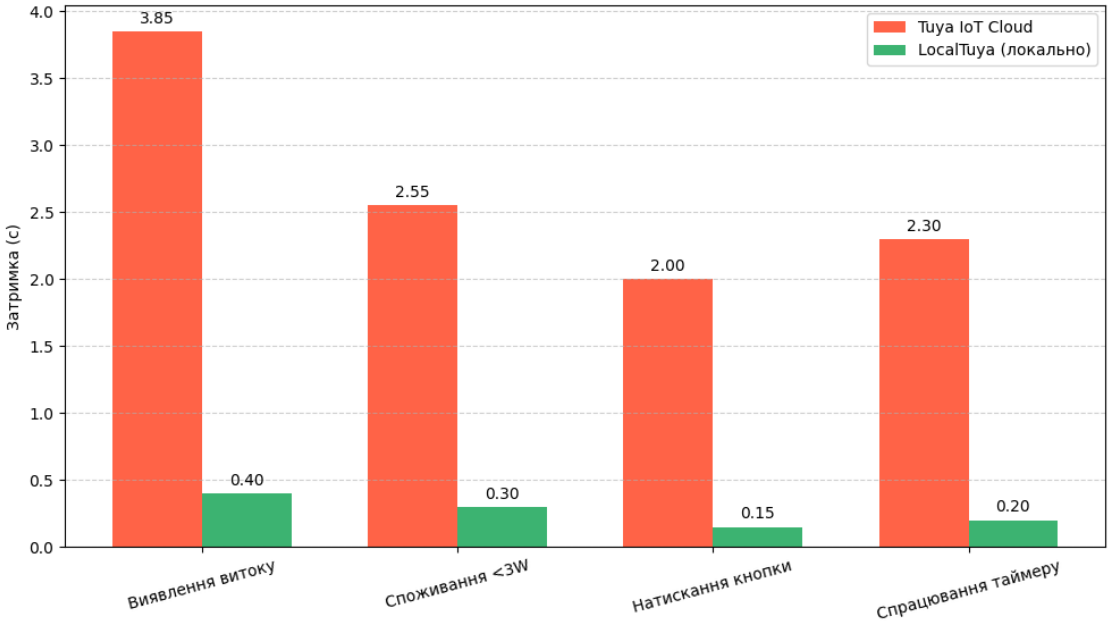

Рисунок 2.25 – Вимірювання затримок виконання сценаріїв

У деяких випадках під час використання хмарної моделі сценарії могли не виконатися взагалі або відпрацьовували з помилками через нестабільність інтернет-з’єднання чи затримку з боку серверів, що особливо критично у випадках, коли автоматизації виконують функції безпеки, де навіть одна секунда може мати важливе значення. Локальне виконання автоматизацій у середовищі Home Assistant дозволяє не тільки скоротити час реакції, а й гарантувати стабільність та точність виконання навіть у складних комбінаціях умов і тригерів. Якщо ж потрібно забезпечити віддалене керування без прив’язки до хмарного сервісу, система може бути доповнена VPN-тунелем, який дозволяє зберегти автономність інфраструктури і водночас надати віддалений доступ без передачі даних на сторонні сервери.

Оцінка стабільності автоматизацій у Home Assistant. Багатоденне тестування стабільності автоматизацій у Home Assistant показало, що незалежно від втрати живлення, перезапусків віртуальної машини чи відсутності Інтернету, жодна з реалізованих автоматизацій не давала збоїв. Після кожного перезапуску Home Assistant автоматично активував усі сценарії, конфігурації яких зберігаються локально у форматі YAML, завдяки чому жодна умова чи тригер не втрачали свою дію. Операційна система Home Assistant, запущена у віртуальному середовищі

VirtualBox, після завантаження одразу забезпечує коректну роботу всіх тригерів, таймерів, логічних операторів і подій, що дозволяє підтримувати систему в повністю працездатному стані.

Окрему увагу під час тестування було приділено реакції системи на втрату доступності окремих пристроїв, зокрема датчиків чи розеток, що були тимчасово знеструмлені або вимкнені. У цих випадках Home Assistant коректно відображає статус пристроїв як «недоступні», при цьому жодна з автоматизацій, що базується на цих пристроях, не активується хибно. Така поведінка запобігає небажаним або помилковим діям. Після відновлення живлення пристрої автоматично з'являються в системі без потреби додаткового налаштування, а відповідні сценарії продовжують працювати в повному обсязі. Інтерфейс Home Assistant, як веб-версія, так і мобільний застосунок, зберігає стабільну роботу у локальній мережі без збоїв, дозволяючи продовжити керування навіть без Інтернету. Усе це свідчить про високу надійність та автономність локальної реалізації розумної автоматизації на базі Home Assistant у поєднанні з інтеграцією LocalTuya, що значно перевищує можливості централізованих хмарних рішень за стабільністю, затримкою виконання та загальною контрольованістю.

Висновки щодо зручності, безпеки та продуктивності. Проведене тестування підтвердило ефективність локальної системи керування IoT-об'єктами на основі Home Assistant із використанням наявної інфраструктури пристроїв Tuya та переходом від централізованої хмарної моделі Tuya IoT Cloud до локальної інтеграції через доповнення LocalTuya, що у сукупності дозволило досягти:

— зручності у користуванні;

— мінімізації затримок;

— підвищення рівня безпеки;

— підвищення стабільності.

Завдяки відкритому інтерфейсу Home Assistant стало можливим створювати повністю автономну систему управління, яка не потребує постійного підключення до інтернету, при цьому надає можливість реалізовувати складні автоматизації навіть без знання програмування через вбудований візуальний редактор, а у разі

потреби — редагувати конфігурації вручну через YAML, що підвищує гнучкість налаштування під конкретні потреби.

Ізоляція від хмарної платформи значно зменшує ризики витоку особистої інформації, оскільки всі дані залишаються в межах локальної мережі, а реакція на події, зокрема критичні, реалізується практично миттєво завдяки відсутності маршрутизації через зовнішні сервери. Завдяки локальній обробці команд керування забезпечується надійне функціонування системи навіть у випадках втрати з’єднання з інтернетом або неполадок у роботі хмарних сервісів, оскільки все управління, автоматизація, сценарії та відображення інтерфейсів здійснюються безпосередньо в локальній інфраструктурі. Усі сценарії зберігають свою працездатність після перезапуску системи або зникнення живлення, оскільки Home Assistant одразу після запуску активує раніше задані правила автоматизації, а після повторного підключення пристроїв — миттєво відновлює зв’язок із ними без потреби у повторному налаштуванні.

Продуктивність такої системи залишається високою навіть при додаванні нових пристроїв, оскільки архітектура Home Assistant дозволяє масштабувати рішення без погіршення швидкодії. Завдяки цьому можна поступово розширювати інфраструктуру, інтегруючи нові датчики, виконавчі пристрої або логіку автоматизації, не побоюючись втрати стабільності чи зниження чутливості сценаріїв. При цьому додаткова перевага полягає у можливості захищеного віддаленого доступу до локальної системи за допомогою VPN, що дозволяє зберегти повну автономність і контроль над даними без потреби у використанні хмарного посередника.

У підсумку, автономний режим роботи Home Assistant з інтеграцією LocalTuya повністю відповідає початковому завданню дослідження, демонструючи перевагу локального підходу в аспектах швидкодії, стабільності, безпеки та функціональної гнучкості. На практиці така система в умовах домашнього застосування дозволяє досягти високого рівня надійності без компромісів у зручності використання, забезпечуючи реальну ефективність у порівнянні з централізованими хмарними рішеннями.

РОЗДІЛ 3 ОПТИМІЗАЦІЯ, ОЦІНКА ТА НАПРЯМИ РОЗВИТКУ СИСТЕМИ

3.1 Порівняльна оцінка реалізованої системи з хмарною моделлю

Для більш обґрунтованої оцінки ефективності впровадженого рішення на базі Home Assistant у порівнянні з типовим хмарним підходом, який реалізується в більшості побутових систем через платформу Tuya, доцільно провести порівняльний аналіз за основними критеріями, зокрема:

— швидкодією;

— доступним функціоналом;

— стійкістю до відмов;

— гнучкістю налаштувань;

— кастомізацією інтерфейсів;

— загальним рівнем контролю над середовищем управління.

Такий підхід дозволяє сформувати цілісну картину щодо реальної ефективності локального варіанту в порівнянні з хмарною моделлю, а також виявити, у яких випадках саме локальна система матиме перевагу, що особливо важливо при переході на власне керування в умовах домашньої автоматизації [28-29].

Порівняння швидкодії – хмара проти локального сервера. Розглядаючи швидкодію як один із критично важливих параметрів для систем автоматизації, особливо в контексті реалізації сценаріїв безпеки, де затримка реакції може мати серйозні наслідки, спостерігається суттєва різниця між хмарним та локальним підходами. У випадку хмарної моделі передача даних відбувається за складним маршрутом: пристрій – маршрутизатор – хмарний сервер Tuya – логіка в хмарі – мобільний застосунок – зворотній сигнал на пристрій, що не лише створює велику кількість залежностей, а й значно підвищує загальну затримку між подією та реакцією. Натомість локальна система керування, реалізована через Home Assistant із використанням LocalTuya, здійснює передавання команд безпосередньо в межах

локальної мережі, де маршрут скорочується до зв’язки пристрій – локальний сервер – пристрій, що суттєво зменшує затримки і дозволяє реагувати майже миттєво, без необхідності звертання до зовнішніх серверів і втручання інтернет-з’єднання. Внаслідок цього час реакції системи на локальній реалізації в окремих випадках виявився у 6–10 разів коротшим, ніж у стандартної хмарної моделі, що особливо важливо для таких критичних подій, як виявлення витоку газу, води, задимлення або вторгнення, де навіть кілька зайвих секунд можуть мати вирішальне значення (див. рис. 3.1).

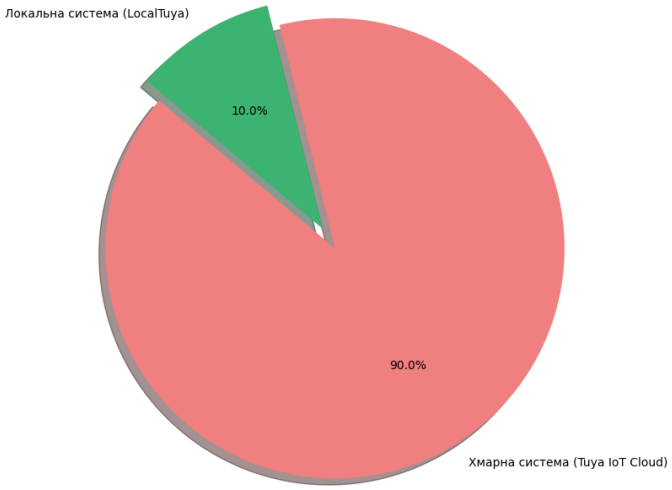

Рисунок 3.1 – Діаграма швидкодії хмарної та локальної систем

Під час практичного тестування також було помітно, що система на базі Home Assistant стабільно реагує на будь-які події без помітної затримки незалежно від навантаження на мережу або тимчасової втрати зовнішнього з’єднання, тоді як хмарна платформа у таких умовах або сповільнює роботу, або взагалі втрачає здатність до управління. У межах щоденного використання така відмінність стає особливо відчутною у сценаріях увімкнення освітлення, кліматичного регулювання чи керування охоронними пристроями, де кожна мить затримки знижує відчуття комфорту й надійності (за наявності таких IoT пристроїв).

Доступність сценаріїв і функціоналу автоматизації. Хмарна модель, яка використовувалась раніше й була побудована на основі платформи Smart Life, хоча

й надає зручний мобільний інтерфейс для створення сценаріїв автоматизації, фактично обмежується базовим функціоналом. Переважна більшість можливостей зводиться до простих умов типу «якщо – тоді», при цьому обмеження стосуються як кількості умов і дій, так і відсутності логічних операторів, лічильників, таймерів, обмежень за часом або сценаріїв із вкладеними залежностями. У результаті навіть для базових сценаріїв часто доводиться створювати декілька окремих автоматизацій для одного й того ж пристрою, а це ускладнює загальне керування й знижує ефективність системи. Відсутність підтримки складної логіки, а також неможливість функціонування без стабільного інтернет-з'єднання ще більше підкреслює обмеження цього підходу. Усі сценарії виконуються виключно на стороні хмари Tuya IoT Cloud, що знижує надійність системи, зокрема у випадках відсутності зв'язку або збою серверів.

На відміну від цього, реалізоване рішення на базі Home Assistant у поєднанні з інтеграцією LocalTuya дозволяє будувати автоматизації практично будь-якої складності. У межах локальної архітектури автоматизації не обмежені за кількістю умов, доступна побудова сценаріїв з будь-якими типами тригерів – станами, атрибутами, подіями, часом тощо. Завдяки використанню умовних блоків, шаблонів, скриптів і зовнішніх сценаріїв з'являється можливість реалізувати гнучку логіку, яка буде адаптована під конкретні потреби, об'єднуючи в собі дані від різних типів сенсорів, зовнішніх джерел інформації та будь-яких інших сутностей Home Assistant. Усе це працює виключно в межах локальної мережі без необхідності виходу в Інтернет, а тому навіть у разі повної втрати зовнішнього зв'язку система зберігає працездатність і виконує автоматизації без змін. Наприклад, у той час як Smart Life дозволяє лише створити сповіщення у випадку витоку води, локальна система на базі Home Assistant може реалізувати сценарій, який при виявленні витоку та відсутності людей вдома одночасно вимикає всі розетки, вмикає сирену, надсилає сповіщення та записує інцидент у журнал. Такий рівень інтеграції та умовності дозволяє створювати справді розумні реакції, які враховують контекст і реагують більш адекватно до ситуації.

Надійність – незалежність від Інтернету, стійкість до збоїв. Якщо порівнювати надійність двох підходів, то хмарна модель на базі Tuya повністю залежить від стабільності з'єднання з Інтернетом та працездатності серверів постачальника. У випадку втрати зв'язку з хмарною інфраструктурою пристрої стають некерованими навіть тоді, коли фізично знаходяться у межах локальної мережі. Також будь-які технічні збої на серверах, зміни політики доступу або впровадження платних тарифів можуть вплинути на функціонування автоматизацій або навіть унеможливити їх виконання. Такий рівень залежності від зовнішніх чинників знижує надійність системи загалом, що вже було підтверджено під час тестування втрати зв'язку або помилок авторизації з боку Tuya IoT Cloud. Натомість реалізоване локальне рішення працює автономно на власному сервері, у даному випадку у вигляді віртуальної машини, яка повністю ізольована від зовнішніх серверів і не потребує постійного доступу до Інтернету. Всі інтерфейси керування, автоматизації, сповіщення та аналітика залишаються доступними навіть за відсутності зовнішнього з'єднання. Крім цього, рішення передбачає можливість встановлення на одноплатні комп'ютери чи компактні міні-ПК, що дозволяє легко масштабувати або переносити систему. За рахунок підтримки функцій резервного копіювання, журналювання та можливості створення знімків системного стану, можна швидко відновити працездатність після збою, оновлення або помилки користувача, що критично важливо для забезпечення безперервності роботи [28].

У результаті вся система демонструє значно вищий рівень автономності, стабільності та передбачуваності в умовах нестабільного інтернет-з'єднання або підвищених вимог до безпеки, що робить локальну реалізацію переважною в сценаріях, де важлива гарантована реакція на події, незалежна від зовнішніх ресурсів чи політик сторонніх компаній.

Вартість – реалізація поточної та майбутньої систем. Оцінювання вартості IoT-системи, реалізованої на базі пристроїв екосистеми Tuya, дозволяє краще зрозуміти її доступність для домашнього використання, адже ці пристрої зазвичай виготовляються у великій кількості, що значно знижує їх собівартість. Основними компонентами стали два розумні розетки (Wi-Fi Smart Socket Plug),

кожен з яких коштує орієнтовно від 150 до 200 грн, що при середньому курсі долара 39 UAH/USD становить приблизно по 5 доларів США за одиницю. У системі використано дві таких розетки, тож їх сумарна вартість дорівнює приблизно 10 USD. Додано також дві розумні лампи (Wi-Fi Smart Bulbs), кожна з яких у середньому коштує близько 225 грн, що еквівалентно приблизно 6 доларам США за штуку — загалом 12 USD за пару. Серед сенсорів присутній датчик витоку води (Wi-Fi Water Leak Sensor), ціна якого варіюється в межах 150–200 грн, тож середній еквівалент — 5 USD. Газовий детектор (Wi-Fi Gas Leak Detector) коштує трохи дорожче — в межах 300–350 грн, що дорівнює приблизно 9 USD. У сумі вся реалізована система обійшлася в близько 36 доларів США без врахування контролера. У якості контролера використовується Home Assistant OS, яка є безкоштовною, і тому не потребує додаткових витрат на ліцензію чи доступ. Проте для тих випадків, коли необхідне окреме апаратне рішення для хостингу цієї платформи, можливо розглянути одноплатний комп’ютер Raspberry Pi 4 Model B з 4GB оперативної пам’яті, вартість якого станом на травень 2025 року складає приблизно 65 USD. У разі придбання Raspberry Pi вартість всієї системи зросте до 101 USD, враховуючи ті ж самі сенсори та пристрої (див. рис. 3.2).

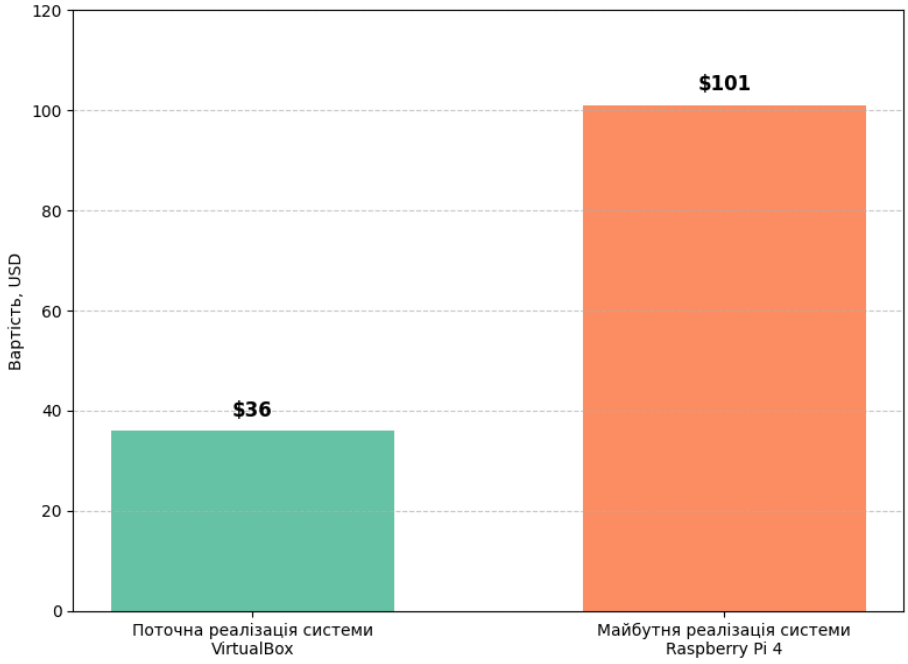

Рисунок 3.2 – Діаграма вартості впровадження IoT-системи

Підсумки щодо гнучкості, керованості та кастомізації. Підсумовуючи ключові аспекти, пов’язані з гнучкістю, керованістю та кастомізацією системи автоматизації на базі Tuya у порівнянні з реалізованим рішенням Home Assistant із використанням LocalTuya, можна констатувати, що платформа Tuya, побудована на основі хмарної моделі Smart Life, має фіксовану структуру, обмежену кількість доступних сценаріїв та жорстко задану логіку, яка не дозволяє вийти за межі запропонованого функціоналу. Усі сценарії реалізуються виключно через мобільний додаток із мінімальною гнучкістю, що унеможливлює реалізацію дійсно складних умов чи сценаріїв з розгалуженою логікою, заснованих на комбінованих подіях, змінних атрибутах або зв’язаних діях між кількома пристроями. Сама система обмежена у керованості через відсутність глибокого логування, повного журналу дій або механізмів резервного відновлення, а також суттєво залежить від стабільності інтернет-з’єднання, оскільки не має локального режиму роботи. Візуальна частина мобільного додатка Tuya також не допускає суттєвих змін: змінити можна лише порядок плиток, що фактично не впливає на організацію управління.

Натомість локальна система, реалізована на основі Home Assistant з інтеграцією LocalTuya, демонструє майже необмежену гнучкість у створенні сценаріїв, які можуть бути побудовані на основі будь-яких подій, станів, атрибутів пристроїв, часових умов або зовнішніх тригерів. Інтеграція дозволяє використовувати умовні оператори, шаблони, цикли, затримки, обмеження, зовнішні скрипти та логіку на основі будь-яких вхідних даних, що забезпечує повноцінний контроль над кожною ланкою в системі. Керування здійснюється не лише через зручний веб-інтерфейс, а й через мобільний застосунок, голосові помічники, REST API, Telegram-боти або зовнішні панелі, а сама платформа дозволяє бачити історію змін, лог подій, поточний стан системи в реальному часі, здійснювати гнучке резервне копіювання і відновлення, автоматизоване тестування сценаріїв та багато інших можливостей [29].

Кастомізація також виходить на якісно новий рівень: можна змінювати зовнішній вигляд панелей, створювати унікальні дашборди під конкретного

користувача, впроваджувати різні теми оформлення, карти, елементи управління та звіти. Наприклад, можна побудувати панель тільки для нічного режиму, тільки для гостей або тільки для смартфона з вертикальним орієнтуванням, що дозволяє персоналізувати систему в деталях, що не лише покращує зручність, а й суттєво підвищує керованість, особливо в умовах різноманітних сценаріїв використання, від простого домашнього середовища до складного інтегрованого офісного простору або віддаленого об’єкта з підвищеними вимогами до надійності.

Таблиця 3.1 – Порівняльна оцінка реалізованої системи з хмарною моделлю

Критерій	**Smart Life (Tuya IoT Cloud)**	**Home Assistant (LocalTuya)**
Швидкодія	Середня (2–4 с)	Висока (0.2–0.5 с)
Сценарії	Обмежені	Гнучкі, багаторівневі
Автономність	Немає	Повна
Кастомізація	Мінімальна	Повна, включно з UI
Надійність	Вразлива до збоїв	Висока
Логування	Немає / обмежене	Повне
Розширюваність	Закрита екосистема	Відкрита платформа

У зведеній таблиці порівняння (див. табл. 3.1) чітко простежується перевага локальної системи за ключовими параметрами: Home Assistant із LocalTuya демонструє високу швидкодію (0.2–0.5 с проти 2–4 с у Tuya), повну автономність, гнучкі багаторівневі сценарії, повну кастомізацію інтерфейсу, стійкість до збоїв, повне логування та розширюваність завдяки відкритій архітектурі. У той час як Tuya має середню швидкодію, обмежену кількість сценаріїв, залежність від хмари, мінімальну кастомізацію, обмежену надійність та замкнуту екосистему.

У такому контексті перехід до локального управління виглядає не просто як технічне вдосконалення, а як стратегічно правильне рішення, що дозволяє усунути:

— обмеження хмарної архітектури;

— зменшити залежність від зовнішніх серверів;

— забезпечити стабільну роботу в умовах нестабільного інтернет-з’єднання.

Також це все дозволяє створити основу для:

— подальшого масштабування;
— інтеграції нових пристроїв;
— підключення до зовнішніх API;
— впровадження енергоефективних алгоритмів;
— підвищення безпеки;
— підвищення загального комфорту.

Можливості, які відкриваються при використанні такої архітектури, по суті є невичерпними і дозволяють будувати автоматизовану інфраструктуру на рівні, що значно перевищує початковий функціонал, закладений у фабричні IoT-рішення.

3.2 Аналіз альтернатив і оцінка вибору платформи

У процесі проєктування спеціалізованої системи керування об’єктами IoT-мереж були розглянуті кілька альтернативних платформ, здатних забезпечити базову та розширену функціональність автоматизації у рамках вже наявної інфраструктури. Пріоритетними критеріями при виборі виступили:

— сумісність із пристроями Tuya;
— можливість реалізації складної автоматизації;
— простота налаштування;
— ресурсна ефективність;
— підтримка розширень;
— інтеграція з іншими сервісами.

Усе це було обумовлено тим, що наявні в системі пристрої базуються на екосистемі Tuya, яка є однією з найбільш масових серед бюджетних IoT-рішень. Альтернативи

на кшталт OpenHAB, Domoticz і Node-RED розглядалися з огляду на те, як кожна з них справляється з конкретними завданнями локального управління, враховуючи обмеження, що накладаються архітектурою хмарної моделі Tuya IoT Cloud, яка мала бути замінена або доповнена більш гнучким рішенням [20, 30-31].

Порівняння Home Assistant з OpenHAB, Domoticz, Node-RED. З погляду технічної реалізації Home Assistant є платформою з відкритим вихідним кодом, яка орієнтована як на домашнє, так і на професійне середовище автоматизації. Вона підтримує широкий спектр інтеграцій завдяки модульній структурі, можливості встановлення сторонніх доповнень через HACS, а також вбудовану підтримку YAML-конфігурацій, яка забезпечує тонке налаштування сценаріїв. Її перевага полягає у поєднанні сучасного веб-інтерфейсу, мобільного застосунку, високої активності спільноти та підтримки локальних сценаріїв автоматизації з можливістю повного резервного копіювання та відновлення системи. Попри те, що деякі аспекти налаштування вимагають часу на вивчення, платформа дозволяє досягти високої гнучкості у створенні автоматизацій, зокрема на базі атрибутів пристроїв, логічних операторів, зовнішніх API або часових умов [30].

У порівнянні з Home Assistant, OpenHAB хоч і має розширену систему правил на основі DSL та підтримку великої кількості протоколів, однак потребує значно вищого рівня підготовки для розгортання і часто вимагає ручного конфігурування всіх компонентів, що ускладнює процес початкового налаштування. Домінуючим недоліком Domoticz виявилася обмеженість у підтримці сучасних інтеграцій, зокрема Tuya, а також примітивний інтерфейс, що не відповідає вимогам гнучкої візуалізації панелей керування. Його можливості щодо сценаріїв автоматизації є дуже базовими, що унеможливлює побудову складних логічних ланцюгів. Node-RED, хоч і пропонує надзвичайну гнучкість у вигляді побудови потоків даних і логіки за допомогою візуального середовища, орієнтований переважно на користувачів зі знанням JavaScript і принципів побудови подієво-орієнтованих моделей, тому не може слугувати повноцінною системою автоматизації у класичному розумінні без значної додаткової конфігурації [30].

Таблиця 3.2 – Порівняння Home Assistant з OpenHAB, Domoticz, Node-RED

Критерій	**Home Assistant**	**OpenHAB**	**Domoticz**	**Node-RED**
Поріг входу	**Низький**	Середній–високий	Низький	Високий
Гнучкість сценаріїв	**Висока (YAML, GUI)**	Висока (rules DSL)	Низька	Дуже висока
Графічний інтерфейс	**Сучасний, адаптивний**	Конфігурований вручну	Примітивний	Побудова блок-схем
Підтримка Tuya	**Через HACS / LocalTuya**	Обмежена	Через MQTT	Через node-red-contrib
Активність спільноти	**Висока**	Висока	Низька	Висока

Порівняння основних характеристик (див. табл. 3.2) дає змогу виділити такі аспекти: Home Assistant має низький поріг входу, високу гнучкість сценаріїв як через графічний редактор, так і через YAML, сучасний інтерфейс, розширену підтримку Tuya через HACS і стабільну активність спільноти. OpenHAB демонструє середньо-високий поріг входу через складність налаштування, подібну гнучкість у правилах автоматизації, але потребує ручного конфігурування і має обмежену підтримку пристроїв Tuya. Domoticz відзначається низьким порогом входу, однак має низьку гнучкість, застарілий інтерфейс і не забезпечує повної інтеграції з сучасними екосистемами. Node-RED має найвищу гнучкість у створенні логіки через блок-схеми, але його використання у ролі основної платформи є утрудненим через високий поріг входу та технічну складність підтримки всієї системи [30].

Підтримка пристроїв Tuya в інших системах. У процесі проєктування системи керування об’єктами IoT-мереж одним із важливих критеріїв стала сумісність платформи з уже наявними пристроями екосистеми Tuya, оскільки їх використання дозволяє знизити витрати на обладнання завдяки широкій доступності, невисокій ціні та великому асортименту. Заміна цих пристроїв на інші

порушила б логіку вже побудованої інфраструктури, тому при виборі платформи необхідно було орієнтуватися на можливість підтримки наявного парку обладнання з перспективою його подальшого розширення [20].

З-поміж усіх платформ, що розглядалися, саме Home Assistant забезпечує найбільш повну та зручну інтеграцію з пристроями Tuya. Офіційна інтеграція реалізована через хмару, де після підключення облікового запису Tuya можливе управління пристроями через додаток або вебінтерфейс. Водночас, існує також можливість локального керування без залучення хмарного сервісу за допомогою стороннього доповнення LocalTuya, яке надається через репозиторій HACS. Дане рішення дає змогу працювати з пристроями без зміни прошивки, зберігаючи їхню гарантію та стабільність роботи. Однак, для коректного функціонування LocalTuya потрібне окреме налаштування кожної сутності вручну, зокрема через ідентифікатори пристроїв, ключі доступу та локальні IP-адреси, які отримуються через інтерфейс розробника платформи Tuya IoT [20].

На відміну від Home Assistant, інші платформи демонструють значні обмеження в контексті підтримки Tuya. OpenHAB може працювати з такими пристроями лише через MQTT-шлюзи або перепрошивку, наприклад на Tasmota, що значно ускладнює процес налаштування і вимагає втручання в прошивку кожного пристрою, з відповідним ризиком втрати стабільності або гарантії. Domoticz у свою чергу має вкрай обмежену офіційну підтримку пристроїв Tuya, а на практиці здебільшого потребує використання сторонніх MQTT-серверів або API-зв’язків, які нестабільні, непередбачувані й не охоплюють повний список моделей. Node-RED теоретично може підтримувати Tuya за рахунок модулів на кшталт "node-red-contrib-tuya-smart-device", однак цей шлях вимагає додаткових знань щодо структури API Tuya, особливостей локального ключа доступу та принципів побудови обробки подій у вигляді потоків, що не завжди виправдано у звичайних сценаріях автоматизації [20, 30].

У підсумку, серед усіх розглянутих платформ тільки Home Assistant дозволяє організувати повноцінну та стабільну роботу з пристроями Tuya без заміни

прошивки, без підключення до хмари та без складних обхідних рішень, зберігаючи при цьому гнучкість і можливість кастомізації сценаріїв автоматизації.

Потреби в ресурсах, складність налаштування. Ще одним критичним фактором при виборі системи автоматизації є ресурсні вимоги, складність інсталяції та придатність до розгортання на слабших пристроях, що особливо актуально у випадках використання застарілого або бюджетного обладнання (див. табл. 3.3).

Таблиця 3.3 – Системні вимоги щодо відкритих платформ керування IoT

Платформа	**Ресурсна вимога (CPU/RAM)**	**Простота встановлення**	**Підходить для слабких пристроїв**
Home Assistant	Середня (1–2 ядра, 2 ГБ)	Дуже проста (образ або Docker)	Так
OpenHAB	Висока (2+ ядра, 2–4 ГБ)	Складна, потребує Java	Обмежено
Domoticz	Низька (1 ядро, 512 МБ)	Проста, але обмежена функціональність	Так
Node-RED	Середня (1–2 ядра, 1 ГБ)	Потребує попередньої підготовки Node.js	Так

Платформи демонструють істотні відмінності за цими критеріями – Home Assistant характеризується середніми вимогами до апаратного забезпечення, потребуючи 1–2 ядра процесора та приблизно 2 ГБ оперативної пам’яті, при цьому має один із найпростіших сценаріїв встановлення — або за допомогою готового образу, або через Docker-контейнер, що робить його зручним для швидкого розгортання в домашньому середовищі. OpenHAB, своєю чергою, потребує більше ресурсів — як мінімум два ядра і 2–4 ГБ ОЗП, при цьому залежить від Java-середовища, що ускладнює початкову інсталяцію й оновлення. Domoticz виділяється найменшими

вимогами до системних ресурсів, працюючи навіть на пристроях із одним ядром процесора та 512 МБ оперативної пам’яті, але при цьому функціональність значно обмежена, а гнучкість автоматизації мінімальна. Node-RED посідає проміжне положення, потребуючи підготовки середовища Node.js, що може викликати труднощі в користувачів без досвіду у сфері веброзробки, хоча його інтерфейс налаштування побудований у вигляді візуальних блок-схем.

З огляду на співвідношення функціональності, вимог до ресурсів і простоти встановлення, саме Home Assistant показує найкращу сумісність з обмеженим апаратним забезпеченням при збереженні широкого спектра можливостей для реалізації складних сценаріїв автоматизації. Платформа може бути встановлена на недорогий одноплатний комп’ютер або старий міні-ПК без використання віртуалізації, використовуючи спеціалізовану операційну систему Home Assistant OS, що дозволяє максимально ефективно задіяти доступні ресурси. Такий підхід робить її універсальним вибором у побутових і напівпрофесійних умовах експлуатації, де критичною є не лише функціональність, а й енергоефективність, автономність і стабільність.

Потенціал інтеграції з іншими сервісами – Telegram, Google Home. У контексті інтеграції з іншими сервісами, такими як Telegram або Google Home, важливою є не лише наявність такої можливості, а й зручність реалізації подібних функцій, оскільки саме завдяки зовнішнім каналам взаємодії автоматизовані системи набувають значно більшої гнучкості й ефективності. Наявність вбудованих механізмів інтеграції з месенджерами, голосовими асистентами, REST API або іншими хмарними інструментами суттєво розширює функціонал та дає змогу створювати як базові, так і складні сценарії керування. У Home Assistant передбачена зручна інтеграція з Telegram, де реалізовані як звичайні повідомлення, так і підтримка інтерактивних кнопок чи обробка введення користувача прямо з чату, що робить можливим повноцінне дистанційне керування без потреби в додаткових панелях чи додатках. Крім того, без жодних ускладнень підтримується Google Home і Alexa, що дозволяє голосом керувати будь-яким підключеним пристроєм, при цьому система підтримує налаштування REST API, Webhook,

MQTT-брокерів та інших відкритих стандартів, які можуть бути використані для створення зв’язків з будь-якими сторонніми сервісами, включно з хмарними або локальними платформами [31].

У випадку OpenHAB наявна інтеграція з Google Home та Alexa, але реалізація її є складною і потребує глибшого розуміння конфігураційних файлів, ручного редагування структур даних та налаштування зовнішніх компонентів. Telegram також підтримується, але лише через окремі модулі, які вимагають ручного додавання та конфігурації правил, що збільшує час налаштування. Попри відкриту архітектуру, яка дає змогу реалізувати гнучкі схеми обміну даними, на практиці вона часто виявляється громіздкою для впровадження на побутовому рівні. Domoticz має ще більше обмежень: підтримка сторонніх сервісів реалізується винятково за допомогою зовнішніх скриптів або додаткових прошивок, сама система не передбачає нативної інтеграції з месенджерами чи голосовими помічниками, а тому можливості розширення виявляються вкрай обмеженими, особливо без досвіду програмування (див. рис. 3.3) [31].

Рисунок 3.3 – Схема інтеграції з Telegram та Google Home

На відміну від цього, Node-RED демонструє вражаючу гнучкість у налаштуванні будь-яких взаємодій через логічні блоки та ноди, включаючи Telegram, Google Assistant, MQTT, WebSocket та інші протоколи, і дозволяє створювати складні сценарії. Проте при всій універсальності система потребує

значних зусиль на етапі конфігурації, оскільки кожен зв’язок реалізується вручну, а візуальна структура проєктів швидко ускладнюється при масштабуванні, що робить таке рішення менш зручним у порівнянні з готовими інтеграціями, передбаченими в Home Assistant [31].

Після порівняння функціональних, технічних і користувацьких характеристик найпопулярніших платформ для автоматизації можна зробити висновок, що саме Home Assistant забезпечує найкращий баланс між простотою налаштування, масштабованістю та можливостями інтеграції з екосистемою Tuya. Локальне керування через доповнення LocalTuya дозволяє уникнути залежності від хмари без необхідності перепрошивання пристроїв, зберігаючи стабільність та контроль над системою. Платформа підтримує прості конфігурації YAML-файлів для новачків і водночас дає розширені інструменти для досвідчених користувачів. Завдяки підтримці широкого кола зовнішніх сервісів, включно з Google Home, Telegram, REST API, MQTT, а також наявності потужної спільноти, що постійно оновлює документацію, створює нові інтеграції та вдосконалює платформу, Home Assistant стає найбільш придатним для практичного використання як у домашніх умовах, так і в офісному середовищі. Інші системи, хоча й мають свої сильні сторони, не забезпечують того рівня універсальності, стабільності та нативної підтримки пристроїв Tuya, якого вимагають сучасні вимоги до локального управління IoT-середовищем, що вказує на обґрунтованість вибору саме Home Assistant як основи автоматизованої інфраструктури [30-31].

3.3 Перспективи вдосконалення побудованої системи

Реалізована система локального управління пристроями екосистеми Tuya на базі Home Assistant вже на етапі впровадження забезпечує необхідний рівень стабільності роботи, гнучкості автоматизації та незалежності від хмарних сервісів, що було однією з головних цілей побудови. Однак з урахуванням стрімкого

розвитку самої платформи, відкритої архітектури та широкої підтримки інтеграцій, існує очевидний потенціал подальшого розширення функціональності, гнучкості та масштабованості цієї системи, а також поглиблення рівня автоматизованого управління пристроями та процесами в межах інтелектуального середовища [32-34].

Розширення функціоналу за рахунок додаткових автоматизацій. Однією з ключових переваг платформи є можливість реалізації розгалужених сценаріїв автоматизації, які базуються на комбінації тригерів, умов та дій у межах YAML-конфігурацій або інтегрованого редактора. У поточній реалізації вже створено низку зручних сценаріїв, що дозволяють реагувати на зміну станів пристроїв, але потенціал значно вищий, зважаючи на те, що система поки що функціонує з обмеженою кількістю пристроїв. При розширенні фізичної інфраструктури можна створити складніші рівні логіки, наприклад, двоступеневі автоматизації на основі показників температури чи вологості, де один сценарій буде активувати обігрівач при низькій температурі, а інший — зволожувач при падінні вологості нижче певного порогу. Окрім цього, актуальним напрямком розвитку є впровадження сповіщень через Telegram-бота, які не лише інформують про зміну стану пристроїв, але й дозволяють взаємодіяти з системою безпосередньо через кнопки в чаті. Варто також реалізувати функції моніторингу та звітності, які б автоматично формували графіки змін параметрів, наприклад, температури або енергоспоживання за тиждень, що стане можливим при підключенні відповідних сенсорів або розумних розеток. За наявності розумного реле, яке контролює все навантаження у приміщенні, можна було б реалізувати глибшу аналітику та адаптивні сценарії енергозбереження з прив’язкою до часу доби, присутності людей або зовнішніх умов [32].

Інтеграція нових типів пристроїв у майбутньому. Незважаючи на те, що система побудована навколо пристроїв екосистеми Tuya, платформа підтримує інтеграцію великої кількості інших типів пристроїв, що відкриває додаткові перспективи. З огляду на актуальні тенденції розвитку локального керування та відкритих стандартів, варто розглядати впровадження пристроїв, які підтримують

такі прошивки як Tasmota або ESPHome. Наприклад, керовані розетки з прошивкою Tasmota забезпечать повністю локальне управління через MQTT-брокер, а сенсори на базі ESPHome ідеально інтегруються в Home Assistant безпосередньо через YAML-конфігурації. Для розширення безпровідної інфраструктури можливе впровадження Zigbee-пристроїв — датчиків руху, температури, освітлення тощо — які працюють автономно без інтернету, але для цього потрібно додати Zigbee-шлюз, який виступає у ролі координатора мережі. Додатково варто звернути увагу на новий стандарт Matter, що базується на локальній IP-комунікації без прив'язки до хмари, забезпечуючи сумісність між пристроями від різних виробників. У майбутньому більшість розумної техніки, ймовірно, буде використовувати саме цей протокол, тож закладення підтримки такого формату вже зараз відкриває довгострокову перспективу масштабування системи (див. рис. 3.4).

Рисунок 3.4 – Схема інтеграції нових типів пристроїв у майбутньому

Інтеграція таких рішень дозволяє сформувати дійсно гетерогенне, але водночас єдине кероване середовище автоматизації, яке не залежить від конкретного виробника чи віддаленої хмарної інфраструктури, забезпечуючи повний контроль, адаптивність до умов та підвищену надійність у роботі. Такий підхід відкриває шлях до створення більш розумного, автономного середовища, в якому користувач має змогу повністю контролювати інфраструктуру без зовнішніх обмежень.

Застосування візуального редактора Node-RED для складніших сценаріїв. Побудована система локального управління пристроями екосистеми Tuya на базі Home Assistant уже на етапі впровадження забезпечує стабільну роботу, гнучкість автоматизації та незалежність від хмарних сервісів, досягнення яких раніше потребувало значних зусиль. Однак з огляду на стрімкий розвиток як самої платформи Home Assistant, так і сумісних рішень, потенціал для її вдосконалення значно перевищує поточний рівень реалізації. Подальше розширення функціональності системи можливе за рахунок впровадження додаткових сценаріїв автоматизації, які дозволяють не лише автоматично керувати базовими діями, а й адаптувати поведінку системи до змін зовнішнього середовища або внутрішніх параметрів. Створення складніших багаторівневих логік, з урахуванням часу, температури, вологості, режимів присутності, історичних даних чи сезонних факторів, може реалізовуватися без потреби програмування вручну, використовуючи вже наявні інструменти (див. рис. 3.5) [33].

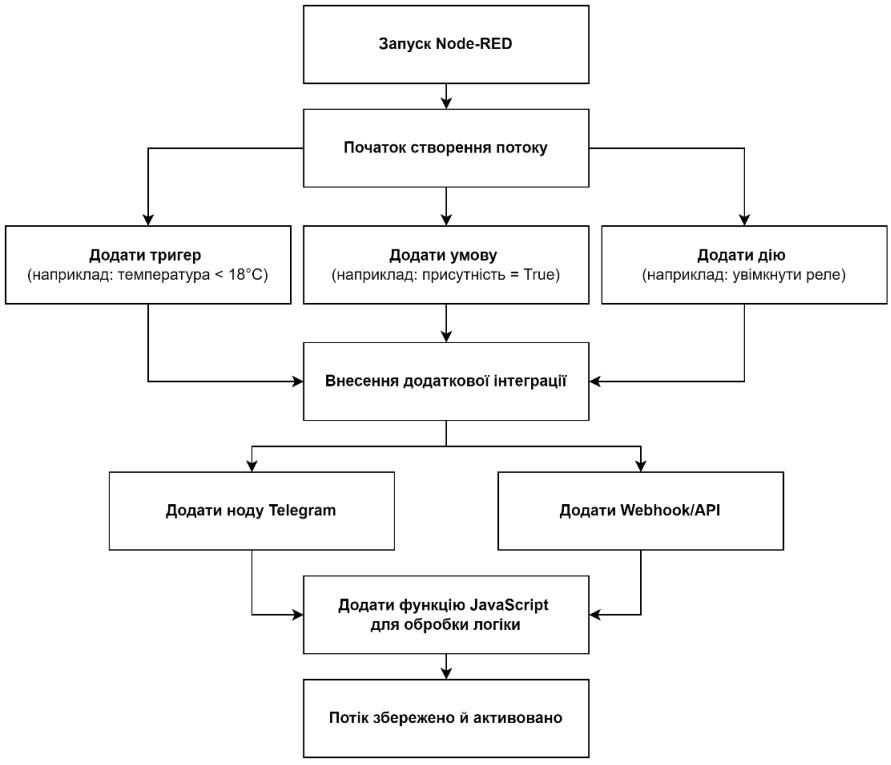

Рисунок 3.5 – Схема автоматизації у Node-RED

Платформа Home Assistant є надзвичайно гнучкою для побудови сценаріїв автоматизації, і навіть поточна реалізація дозволяє поєднувати декілька умов та дій у складні комбіновані схеми, однак реальний потенціал автоматизації розкривається лише за наявності достатньої кількості сенсорів, розумних розеток, реле та інших пристроїв. За умови наявності датчиків температури і вологості можливо налаштувати логіку керування обігрівачами або зволожувачами повітря відповідно до заданих порогів. Додатково інтеграція повідомлень про зміну стану або помилки через Telegram, з підтримкою взаємодії через інлайн-кнопки в чаті, дозволить підвищити інтерактивність і зручність взаємодії. Застосування звітності й аналітики, наприклад у вигляді автоматично створених щотижневих графіків зміни температури чи енергоспоживання, надасть змогу не лише оперативно реагувати, але й аналізувати довгострокові тенденції. Розширення кількості інтегрованих пристроїв, як-то розумні розетки, реле чи сенсори з підтримкою енергомоніторингу, значно підвищить рівень контролю над системою [30-31, 33].

Водночас, платформа не обмежується виключно пристроями екосистеми Tuya. З урахуванням зростання популярності локальних протоколів і відкритих стандартів зростає й доцільність інтеграції альтернативних пристроїв. Пристрої на базі прошивок Tasmota або ESPHome дозволяють створювати повністю локальне управління без використання хмар, у тому числі за допомогою протоколу MQTT, який є базовим у багатьох розумних системах. Впровадження Zigbee-пристроїв, таких як датчики руху, освітлення, температури та кнопки, забезпечить можливість створення гнучкої безпровідної мережі автоматизації навіть у разі відсутності доступу до інтернету. Для цього буде потрібен Zigbee-шлюз, однак в обмін отримується стабільність, швидкодія та відсутність залежності від зовнішніх сервісів. Додатково варто врахувати потенціал новітнього протоколу Matter, який забезпечує локальну взаємодію та очікувано стане основою для більшості нових пристроїв найближчими роками. Таким чином, поступове доповнення системи пристроями з різними протоколами дозволить побудувати єдину гнучку інфраструктуру без жорсткої прив’язки до конкретного виробника чи хмарної платформи.

Для реалізації ще складніших сценаріїв автоматизації доцільним буде використання візуального середовища Node-RED, що дозволяє створювати логіку за допомогою блокових схем. На відміну від YAML-редактора Home Assistant, де складні зв’язки між подіями й умовами можуть швидко ускладнюватися, Node-RED дає змогу наочно структурувати автоматизацію, знижуючи ймовірність помилок. При цьому інтеграція з Home Assistant здійснюється через відповідні ноди, які забезпечують повний доступ до сутностей, подій, атрибутів і сервісів платформи. Обробка зовнішніх API, прийом Webhook-запитів, взаємодія з Telegram, погодними сервісами, сторонніми пристроями або базами даних відбувається значно зручніше у такому середовищі. Додатково підтримується використання JavaScript для побудови складної логіки, що є особливо актуальним для систем, які виходять за межі побутового застосування та потребують взаємодії між кількома платформами чи передавання подій між різними контролерами [30-31, 33].

У підсумку, реалізована система автоматизації на основі Home Assistant має чіткий потенціал для вдосконалення через:

— впровадження більш глибокої автоматизації;

— розширення переліку пристроїв;

— використання візуального середовища Node-RED;

— перехід до повністю автономної архітектури.

Усі ці напрямки дають змогу створити більш масштабовану, гнучку та надійну систему керування пристроями, яка відповідатиме сучасним вимогам до приватності, стабільності, доступності й гнучкої адаптації до змін середовища у найближчому майбутньому.

ВИСНОВКИ

У ході реалізації було досліджено роботу екосистеми Tuya – мобільної платформи Smart Life, що дало змогу оцінити її базову функціональність, простоту налаштування та основні обмеження. Незважаючи на зручний інтерфейс, система виявилась залежною від хмарних сервісів та стабільного інтернету, що створює ризики для автономної роботи й обмежує можливості складніших сценаріїв автоматизації.

Після розгортання Home Assistant та налаштування інтеграції LocalTuya пристрої отримали можливість працювати у повністю локальному режимі без втрати контролю, що дозволило знизити затримки реакції, підвищити стабільність і забезпечити повну незалежність від зовнішніх сервісів. Розміщення системи на локальному сервері також дало змогу зберігати історію подій, підтримувати контроль через локальну мережу та виконувати оновлення з мінімальними ризиками.

На основі доступних інструментів Home Assistant були реалізовані комбіновані сценарії, що поєднують кілька умов, тригерів і дій, без потреби в зовнішніх інструментах програмування. Усе це дозволило створити ефективні автоматизації для щоденних завдань, з урахуванням локальної обробки подій, доступності інтерфейсу та адаптивності до змін у конфігурації пристроїв.

У підсумку, перехід на Home Assistant не лише забезпечив стабільнішу та безпечнішу роботу системи, а й відкрив нові можливості для її розширення та адаптації до складніших потреб. Локальне управління, широка підтримка інтеграцій і незалежність від хмарних сервісів роблять цю платформу оптимальним вибором для побудови надійного середовища керування IoT-пристроями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Top IoT Applications in Agriculture, Healthcare, Smart Homes & Beyond. EICTA IIT Kanpur. URL: https://eicta.iitk.ac.in/knowledge-hub/internet-of-things/iot-applications-smart-homes-healthcare-agriculture-and-industrial-iot/ (дата звернення: 25.02.2025)
2. Benefits of Smart Home Automation. TechTarget. URL: https://www.techtarget.com/iotagenda/definition/smart-home-or-building (дата звернення: 27.02.2025)
3. Tuya Smart: Building an IoT Developer Ecosystem. Tuya Global. URL: https://www.tuya.com/news-details/tuya-smart-iot-ecosystem (дата звернення: 01.03.2025)
4. Limitations of Closed IoT Platforms. IoT For All. URL: https://www.iotforall.com/closed-vs-open-iot-platforms (дата звернення: 03.03.2025)
5. Architecture of an IoT System. GeeksforGeeks. URL: https://www.geeksforgeeks.org/architecture-of-internet-of-things-iot/ (дата звернення: 06.03.2025)
6. Types of IoT Devices. IoT Agenda. URL: https://www.techtarget.com/iotagenda/definition/IoT-device (дата звернення: 08.03.2025)
7. IoT Communication Protocols. IoT Analytics. URL: https://iot-analytics.com/list-of-iot-protocols/ (дата звернення: 12.03.2025)
8. Centralized vs Decentralized IoT Systems. Datafloq. URL: https://datafloq.com/read/centralized-vs-decentralized-iot-architecture/ (дата звернення: 15.03.2025)
9. Tuya Cloud Architecture Explained. Tuya Developer. URL: https://developer.tuya.com/en/docs/iot/tuya-cloud?id=K95ztz7ijv7f7 (дата звернення: 19.03.2025)
10. Smart Life App User Guide. Tuya Support. URL: https://support.tuya.com/en/help/_id/457517687 (дата звернення: 22.03.2025)
11. How Smart Home Automation Works. How-To Geek. URL: https://www.howtogeek.com/789784/how-does-smart-home-automation-work/ (дата звернення: 25.03.2025)
12. Cloud vs Local Control in Smart Homes. The Ambient. URL: https://www.the-ambient.com/guides/cloud-vs-local-control-smart-home-2650 (дата звернення: 27.03.2025)
13. Examples of Tuya-Compatible Devices. Zigbee Alliance Blog. URL: https://zigbeealliance.org/news_and_articles/tuya-zigbee-device-types/ (дата звернення: 30.03.2025)
14. Smart Life Automation Tutorial. SmartHomeBeginner. URL: https://www.smarthomebeginner.com/smart-life-automation-guide/ (дата звернення: 02.04.2025)

15. Limitations of Tuya App Automation. Home Assistant Community. URL: https://community.home-assistant.io/t/limitations-of-tuya-integration/ (дата звернення: 03.04.2025)
16. Understanding IoT Topologies. Network World. URL: https://www.networkworld.com/article/3327300/iot-network-topologies.html (дата звернення: 07.04.2025)
17. Tuya Smart Limitations. Reddit – r/homeassistant. URL: https://www.reddit.com/r/homeassistant/comments/q4z7gu/tuya_limitations/ (дата звернення: 11.04.2025)
18. What Is LocalTuya? HACS Docs. URL: https://hacs.xyz/docs/integration/localtuya/ (дата звернення: 13.04.2025)
19. Why Move from Cloud to Local in Smart Homes (2023). Home Assistant Blog. URL: https://www.home-assistant.io/blog/2023/02/18/local-smart-home-benefits/ (дата звернення: 17.04.2025)
20. Benefits of Offline Automation. SmarterHome. URL: https://www.smarterhome.io/articles/autonomous-home-automation (дата звернення: 23.04.2025)
21. Compatible Devices with Tuya and Home Assistant. Home Assistant Docs. URL: https://www.home-assistant.io/integrations/tuya/ (дата звернення: 26.04.2025)
22. Common Smart Home Scenarios. BuildWithAutomation. URL: https://buildwithautomation.com/common-smart-home-scenarios/ (дата звернення: 27.04.2025)
23. Cloud Management Drawbacks. Smart Home News. URL: https://www.smarthomenews.org/cloud-vs-local-pros-cons (дата звернення: 30.04.2025)
24. Tuya Cloud Integration Setup. Tuya Developer Portal. URL: https://developer.tuya.com/en/docs/iot/Cloud_Project?id=K95ztz7ijv7f7 (дата звернення: 01.05.2025)
25. LocalTuya Integration Guide. GitHub – LocalTuya. URL: https://github.com/rospogrigio/localtuya (дата звернення: 03.05.2025)
26. Home Assistant Local Network Control. Home Assistant Community. URL: https://community.home-assistant.io/t/tuya-local-control/ (дата звернення: 03.05.2025)
27. Introduction to Lovelace UI and YAML. Home Assistant Docs. URL: https://www.home-assistant.io/lovelace/ (дата звернення: 04.05.2025)
28. Using Device IDs in LocalTuya. Home Assistant Forum. URL: https://community.home-assistant.io/t/how-to-find-tuya-device-id/ (дата звернення: 04.05.2025)
29. Resilience of Local Smart Home Systems. The Ambient. URL: https://www.the-ambient.com/guides/local-smart-home-survival-guide-2645 (дата звернення: 05.05.2025)
30. Customization in Home Assistant vs Tuya. HomeTechWorld. URL: https://www.hometechworld.com/home-assistant-vs-tuya/ (дата звернення: 05.05.2025)
31. Home Assistant vs OpenHAB vs Domoticz vs Node-RED. MakeUseOf. URL: https://www.makeuseof.com/home-automation-platforms-compared/ (дата звернення: 05.05.2025)

32. Smart Home Integrations – Telegram, Google. Home Assistant Integrations. URL: https://www.home-assistant.io/integrations/ (дата звернення: 05.05.2025)

33. Creating Advanced Automations in HA. Home Assistant Docs. URL: https://www.home-assistant.io/docs/automation/editor/ (дата звернення: 05.05.2025)

34. Node-RED with Home Assistant. Node-RED Docs. URL: https://nodered.org/docs/user-guide/hass-integration (дата звернення: 05.05.2025)

35. Building Fully Offline Smart Homes. SmarterHome. URL: https://www.smarterhome.io/articles/offline-smart-home (дата звернення: 05.05.2025)

36. Терехов О.С., Бондарчук А.П. Програмна реалізація локального керування пристроями Tuya в середовищі Home Assistant. «Застосування програмного забезпечення в ІКТ» : Всеукр. науково-техн. конф., м.Київ, 24 квітня 2025 р. С.535-538.

37. Терехов О.С., Бондарчук А.П. Інтелектуальна модель локального керування об’єктами IoT-мереж у системах автоматизації. «Сучасні інтелектуальні інформаційні технології в науці та освіті» : V Всеукр. науково-практ. конф., м.Київ, 15 травня 2025 р. (подано до публікації).

Додаток А

Конфігурація дашборда в Home Assistant для керування побутовими пристроями

```yaml
title: Розумний дім
views:
  - title: Загальне
    path: overview
    type: custom:layout-card
    layout: masonry
    icon: mdi:home-assistant
    cards:
      - type: horizontal-stack
        cards:
          - type: button
            name: Ванна
            icon: mdi:shower
            tap_action:
              action: navigate
              navigation_path: /dashboard-test/bath
          - type: button
            name: Вітальня
            icon: mdi:sofa
            tap_action:
              action: navigate
              navigation_path: /dashboard-test/livingroom
          - type: button
            name: Кухня
            icon: mdi:fridge
            tap_action:
              action: navigate
              navigation_path: /dashboard-test/kitchen
          - type: button
            name: Спальня
            icon: mdi:bed
            tap_action:
              action: navigate
              navigation_path: /dashboard-test/bedroom
          - type: button
            name: Освітлення
            icon: mdi:lightbulb-group
            tap_action:
              action: navigate
              navigation_path: /dashboard-test/lights
          - type: button
            name: Розетки
            icon: mdi:power-socket-eu
            tap_action:
              action: navigate
              navigation_path: /dashboard-test/sockets
          - type: button
            name: Датчики
            icon: mdi:chip
            tap_action:
              action: navigate
              navigation_path: /dashboard-test/sensors
      - type: custom:mushroom-person-card
        entity: person.toleksandr
        name: Oleksandr Terekhov
        icon_type: entity-picture
      - type: weather-forecast
        entity: weather.forecast_golovna
        name: Погода
  - title: Вітальня
    path: livingroom
    icon: mdi:sofa
    type: custom:layout-card
    layout: masonry
    cards:
      - type: custom:mushroom-light-card
        entity: light.corridor_light
        use_light_color: true
        show_brightness_control: true
        show_color_temp_control: true
        show_color_control: true
        name: Коридор
        icon_color: amber
```

```yaml
- title: Кухня
  path: kitchen
  icon: mdi:fridge
  type: custom:layout-card
  layout: masonry
  cards:
    - type: custom:mushroom-media-player-card
      entity: media_player.mitv_mssp0
      name: Mi TV
    - type: custom:mushroom-cover-card
      entity: switch.kettle_socket_1
      name: Чайник
    - type: custom:mini-graph-card
      name: Потужність чайника
      entities:
        - sensor.kettle_power
      line_color: orange
      hours_to_show: 24
      points_per_hour: 2
      show:
        fill: true
    - type: custom:mini-graph-card
      name: Напруга/Струм чайника
      entities:
        - sensor.kettle_voltage
        - sensor.kettle_current
      hours_to_show: 24
      points_per_hour: 2
    - type: custom:mushroom-entity-card
      entity: binary_sensor.gas_detector_gas
      name: Детектор газу
      icon_color: red
    - type: custom:mushroom-entity-card
      entity: sensor.gas_detector_gas
      name: Концентрація газу
- title: Ванна
  path: bath
  icon: mdi:shower
  type: custom:layout-card
  layout: masonry
  cards:
    - type: custom:mushroom-entity-card
      entity: binary_sensor.bath_water_leak_sensor_moisture
      name: Протікання
      icon_color: blue
- title: Спальня
  path: bedroom
  icon: mdi:bed
  type: custom:layout-card
  layout: masonry
  cards:
    - type: custom:mushroom-cover-card
      entity: switch.charger_socket_socket_1
      name: Зарядка
    - type: custom:mushroom-light-card
      entity: light.room_light
      use_light_color: true
      show_brightness_control: true
      show_color_temp_control: true
      show_color_control: true
      name: Освітлення кімнати
      icon_color: amber
    - type: custom:mushroom-media-player-card
      entity: media_player.room_tv
      name: Телевізор
    - type: custom:mini-graph-card
      name: Потужність зарядки
      entities:
        - sensor.charger_socket_power
      line_color: teal
      hours_to_show: 24
    - type: custom:mini-graph-card
      name: Напруга/Струм зарядки
      entities:
        - sensor.charger_socket_voltage
        - sensor.charger_socket_current
```

```
      hours_to_show: 24
- title: Освітлення
  path: lights
  icon: mdi:lightbulb-group
  type: custom:layout-card
  layout: masonry
  cards:
    - type: custom:mushroom-light-card
      entity: light.corridor_light
      use_light_color: true
      show_brightness_control: true
      show_color_temp_control: true
      show_color_control: true
      name: Вітальня
    - type: custom:mushroom-light-card
      entity: light.room_light
      use_light_color: true
      show_brightness_control: true
      show_color_temp_control: true
      show_color_control: true
      name: Спальня
- title: Розетки
  path: sockets
  icon: mdi:power-socket-eu
  type: custom:layout-card
  layout: masonry
  cards:
    - type: custom:mushroom-cover-card
      entity: switch.kettle_socket_1
      name: Чайник
    - type: custom:mushroom-cover-card
      entity: switch.charger_socket_socket_1
      name: Зарядка
- title: Датчики
  path: sensors
  icon: mdi:chip
  type: custom:layout-card
  layout: masonry
  cards:
    - type: custom:mushroom-entity-card
      entity: sensor.kettle_power
      name: Потужність Чайника
    - type: custom:mushroom-entity-card
      entity: sensor.kettle_voltage
      name: Напруга Чайника
    - type: custom:mushroom-entity-card
      entity: sensor.kettle_current
      name: Струм Чайника
    - type: custom:mushroom-entity-card
      entity: sensor.charger_socket_power
      name: Потужність Зарядки
    - type: custom:mushroom-entity-card
      entity: sensor.charger_socket_voltage
      name: Напруга Зарядки
    - type: custom:mushroom-entity-card
      entity: sensor.charger_socket_current
      name: Струм Зарядки
    - type: custom:mushroom-entity-card
      entity: binary_sensor.gas_detector_gas
      name: Газ Детектор
    - type: custom:mushroom-entity-card
      entity: sensor.gas_detector_gas
      name: Концентрація Газу
    - type: custom:mushroom-entity-card
      entity: binary_sensor.bath_water_leak_sensor_moisture
      name: Датчик Витоку Води
```

Додаток Б

Комбінований сценарій автоматизації у Home Assistant: автокерування при вході/виході з дому з перевіркою часу

```yaml
alias: Автокерування при вході/виході з дому з перевіркою часу
description: >-
  Вмикає розетки завжди, світло — тільки після заходу; вимикає все після виходу
  з дому з затримкою
triggers:
  - entity_id: person.toleksandr
    from: home
    to: not_home
    id: left_home
    trigger: state
  - entity_id: person.toleksandr
    from: not_home
    to: home
    id: arrived_home
    trigger: state
conditions: []
actions:
  - choose:
      - conditions:
          - condition: trigger
            id: arrived_home
        sequence:
          - target:
              entity_id:
                - switch.kettle_socket_1
                - switch.charger socket socket 1
            action: switch.turn_on
            data: {}
          - condition: sun
            after: sunset
          - target:
              entity_id:
                - light.room light
                - light.corridor_light
            action: light.turn_on
            data: {}
      - conditions:
          - condition: trigger
            id: left_home
        sequence:
          - delay:
              minutes: 5
          - condition: state
            entity_id: person.toleksandr
            state: not_home
          - target:
              entity_id:
                - switch.kettle_socket_1
                - switch.charger socket socket 1
            action: switch.turn_off
            data: {}
          - target:
              entity_id:
                - light.room light
                - light.corridor light
            action: light.turn_off
            data: {}
mode: single
```

Додаток В

Комбінований сценарій автоматизації у Home Assistant: автоматизація світла ввечері та вночі

```yaml
alias: Автоматизація світла ввечері та вночі
description: Увімкнення після заходу сонця, вимкнення о 00:30
triggers:
  - event: sunset
    id: sunset_trigger
    trigger: sun
  - at: "00:30:00"
    id: midnight_trigger
    trigger: time
conditions: []
actions:
  - choose:
      - conditions:
          - condition: trigger
            id: sunset_trigger
        sequence:
          - target:
              entity_id:
                - light.corridor_light
                - light.room light
            action: light.turn_on
            data: {}
      - conditions:
          - condition: trigger
            id: midnight_trigger
        sequence:
          - target:
              entity_id:
                - light.corridor light
                - light.room light
            action: light.turn_off
            data: {}
mode: single
```

Додаток Г

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

НА ТЕМУ: «МОДЕЛЬ СПЕЦІАЛІЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ОБ'ЄКТАМИ ІОТ МЕРЕЖ»

Виконав студент групи ІСД-41
Терехов Олександр Сергійович
Керівник дипломної роботи:
Бондарчук Андрій Петрович,
д-р техн. наук, проф.

Київ – 2025

Мета, об’єкт, предмет дослідження

Мета роботи – спрощення локального управління пристроями IoT-мереж за рахунок використання принципів автономної інтеграції та автоматизації

Об’єкт дослідження – процес локальної взаємодії та керування побутовими IoT-пристроями в інфраструктурі розумного будинку

Предмет дослідження – засоби локальної інтеграції та автоматизації з використанням відкритих протоколів і платформ

Актуальність роботи

У роботі розглянуто можливості платформи Home Assistant для автономного керування Wi-Fi-пристроями екосистеми Tuya. Проаналізовано два підходи інтеграції: хмарний (Tuya IoT Cloud та Smart Life) та локальний (Home Assistant та LocalTuya), із порівнянням їхньої ефективності, затримки реакції та залежності від інтернет-з’єднання.

Реалізовано комбіновані сценарії автоматизації керування освітленням та енергоспоживанням, що дозволило оцінити стабільність та зручність локального управління. Визначено переваги локального підходу, зокрема автономність, швидкодію та безпеку при мінімальній складності впровадження.

Попередня архітектура на базі Tuya Smart Life

Рис. 1.10 – Зовнішній вигляд наявних керованих пристроїв

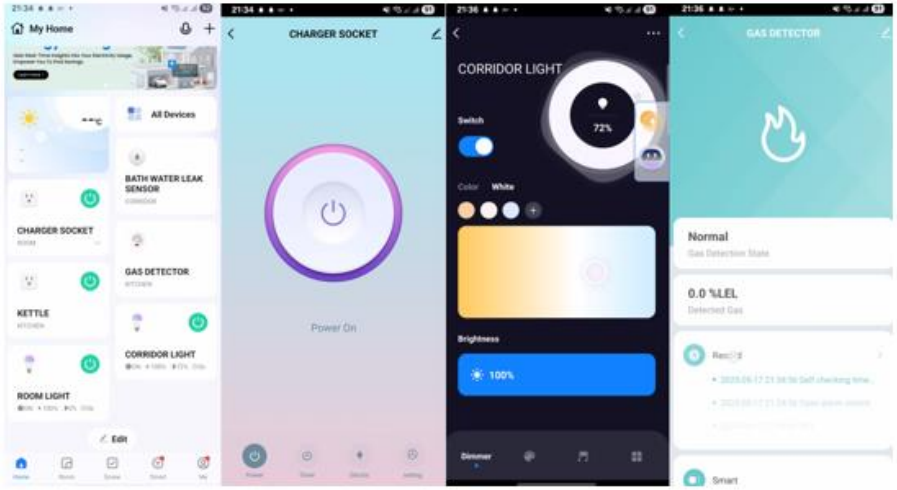

Рис. 1.5 – Інтерфейс застосунку Smart Life для керування пристроями

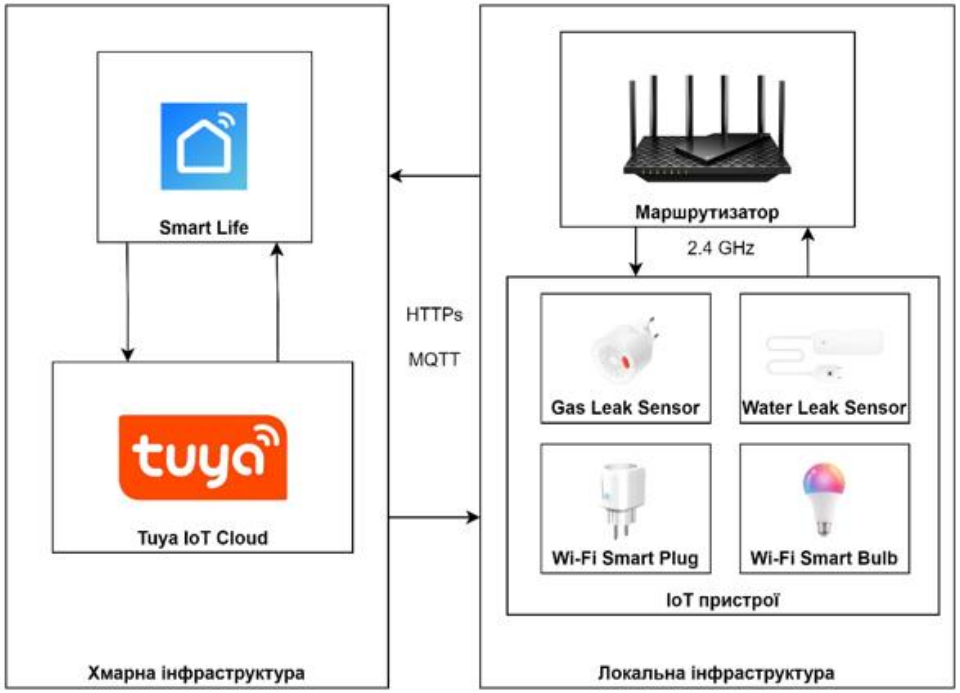

Рис. 1.16 – Топологія поточної хмарної системи

Проблематика існуючої реалізації

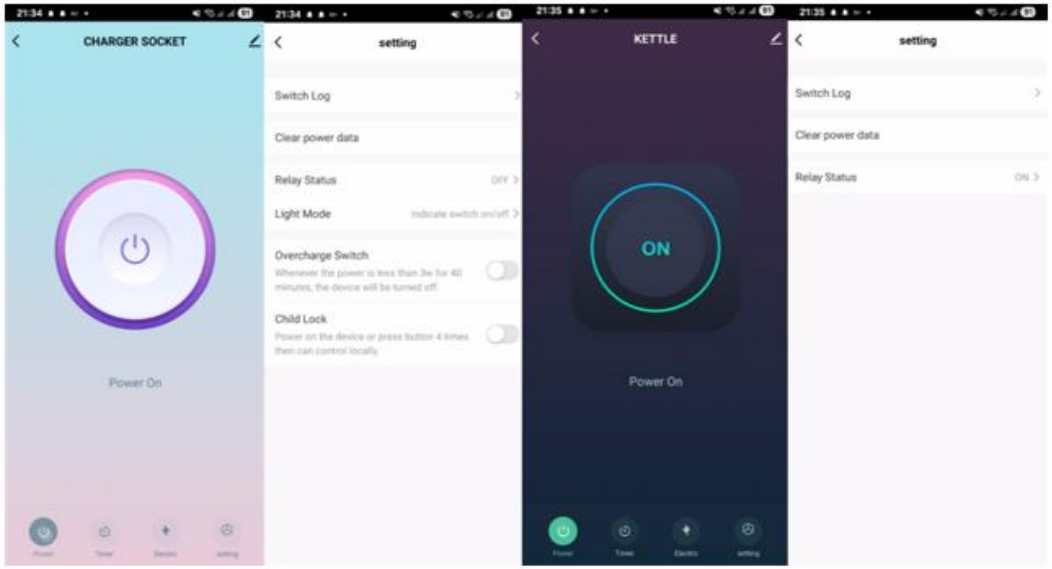

Рис. 1.8 – Відмінність у інтерфейсі та функціоналі однакових IoT-пристроїв

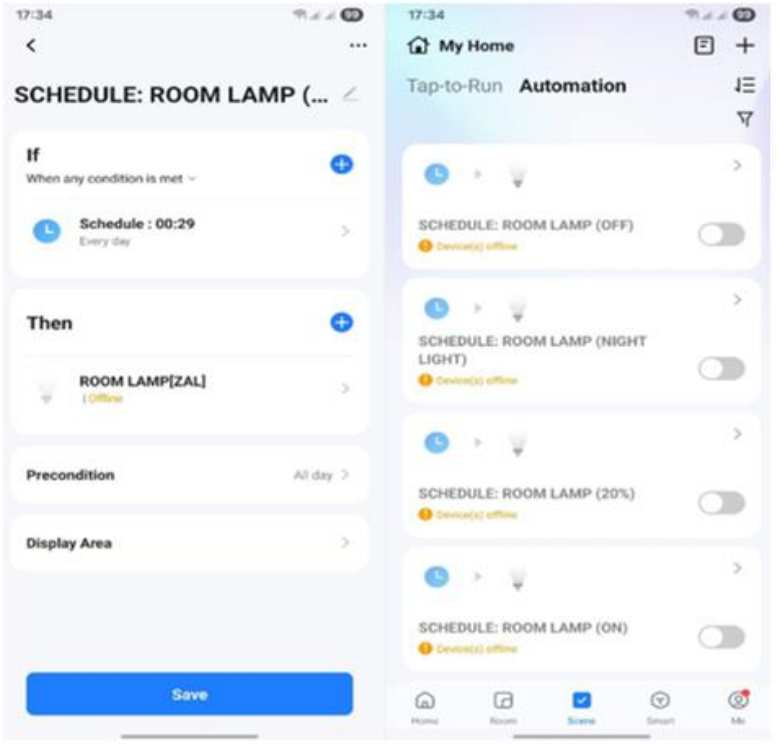

Рис. 1.9 – Відсутність виконання налаштованих автоматизації у разі втрати інтернет-зв'язку

Архітектура запропонованого рішення: Home Assistant та інтеграція LocalTuya

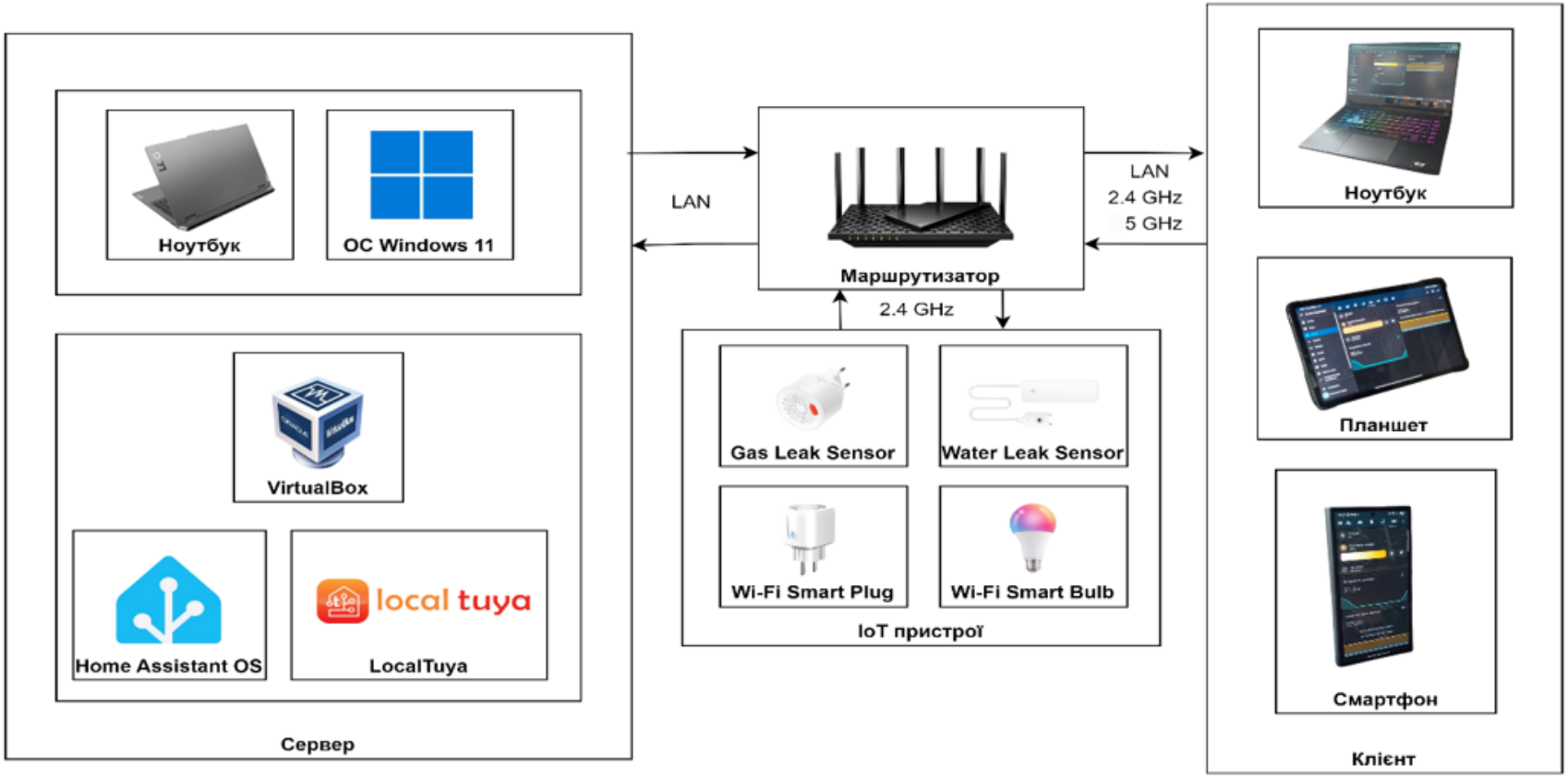

Рис. 2.2 – Архітектура моделі спеціалізованої системи управління на базі Home Assistant

Розгортання системи на базі Home Assistant OS (VirtualBox)

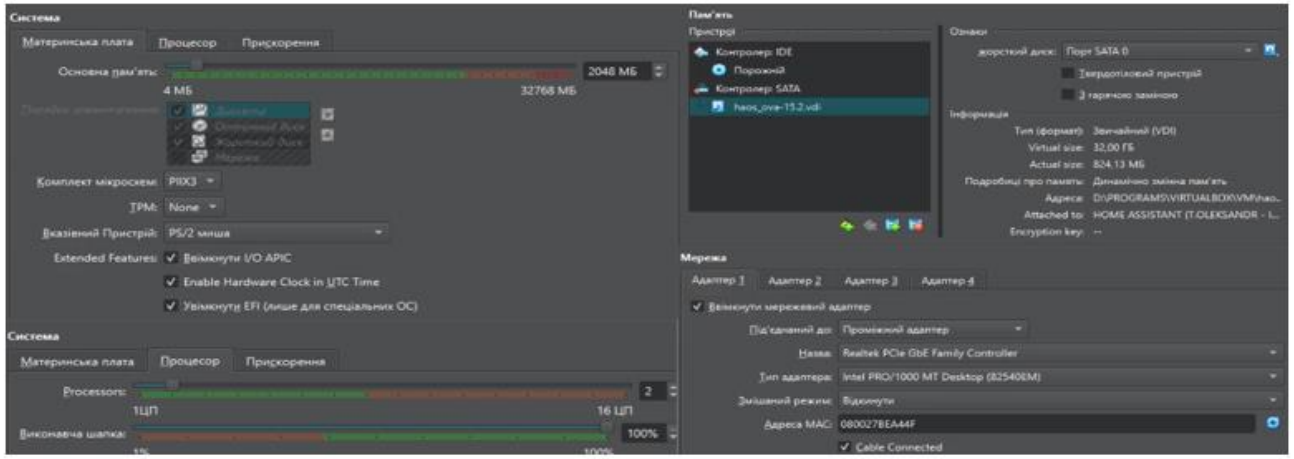

Рис. 2.3 – Конфігурація віртуалізованого середовища VirtualBox

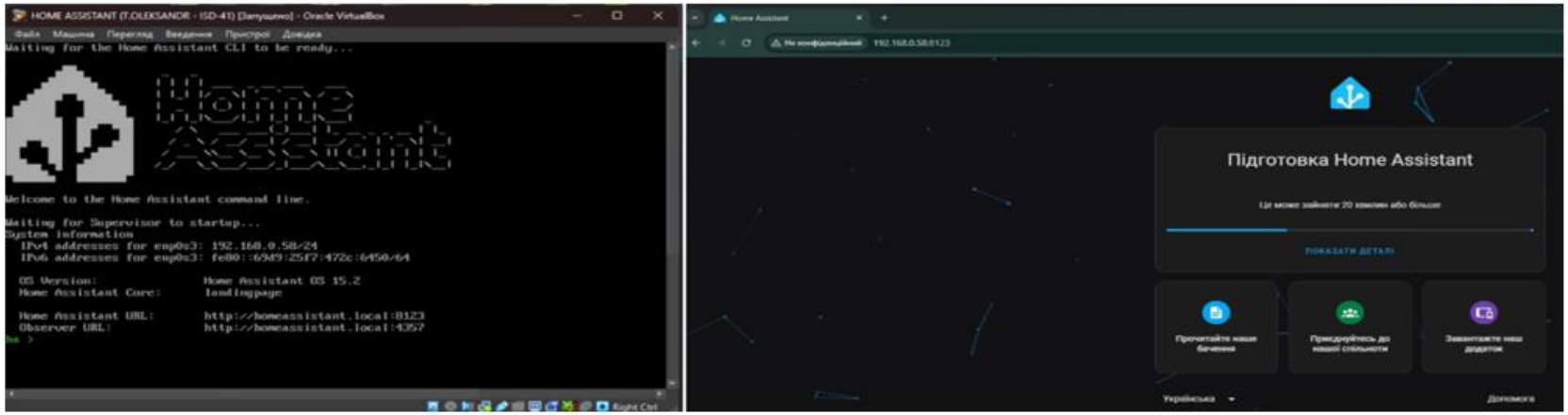

Рис. 2.4 – Ініціалізація та підготовка Home Assistant OS

Інтеграція існуючих пристроїв: Tuya та LocalTuya

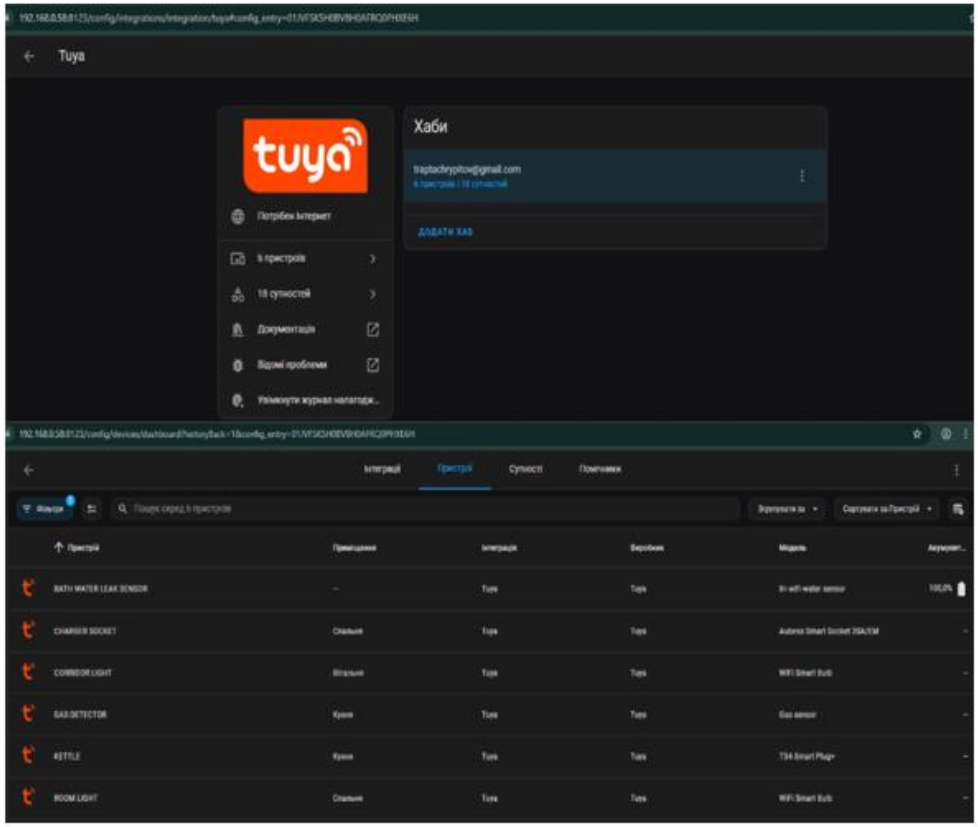

Рис. 2.7 – Хмарна інтеграція пристроїв Tuya з Home Assistant

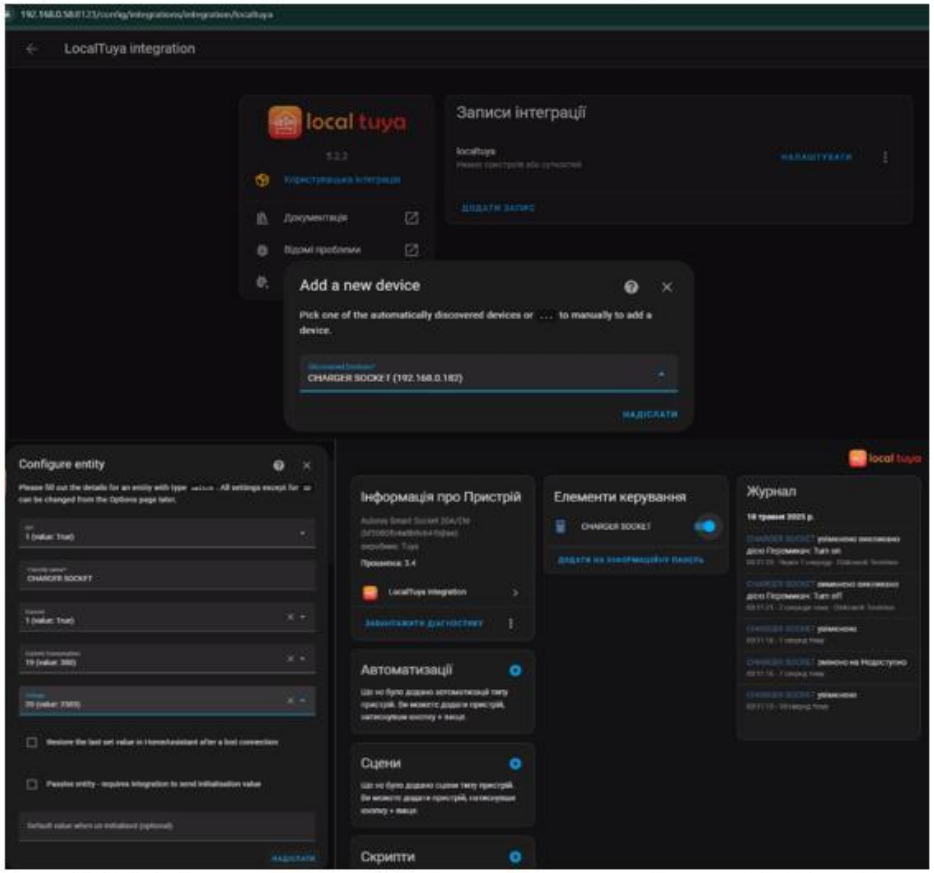

Рис. 2.15 – Підключення пристроїв через локальну інтеграцію LocalTuya

Згенерований дашборд користувача після хмарної інтеграції

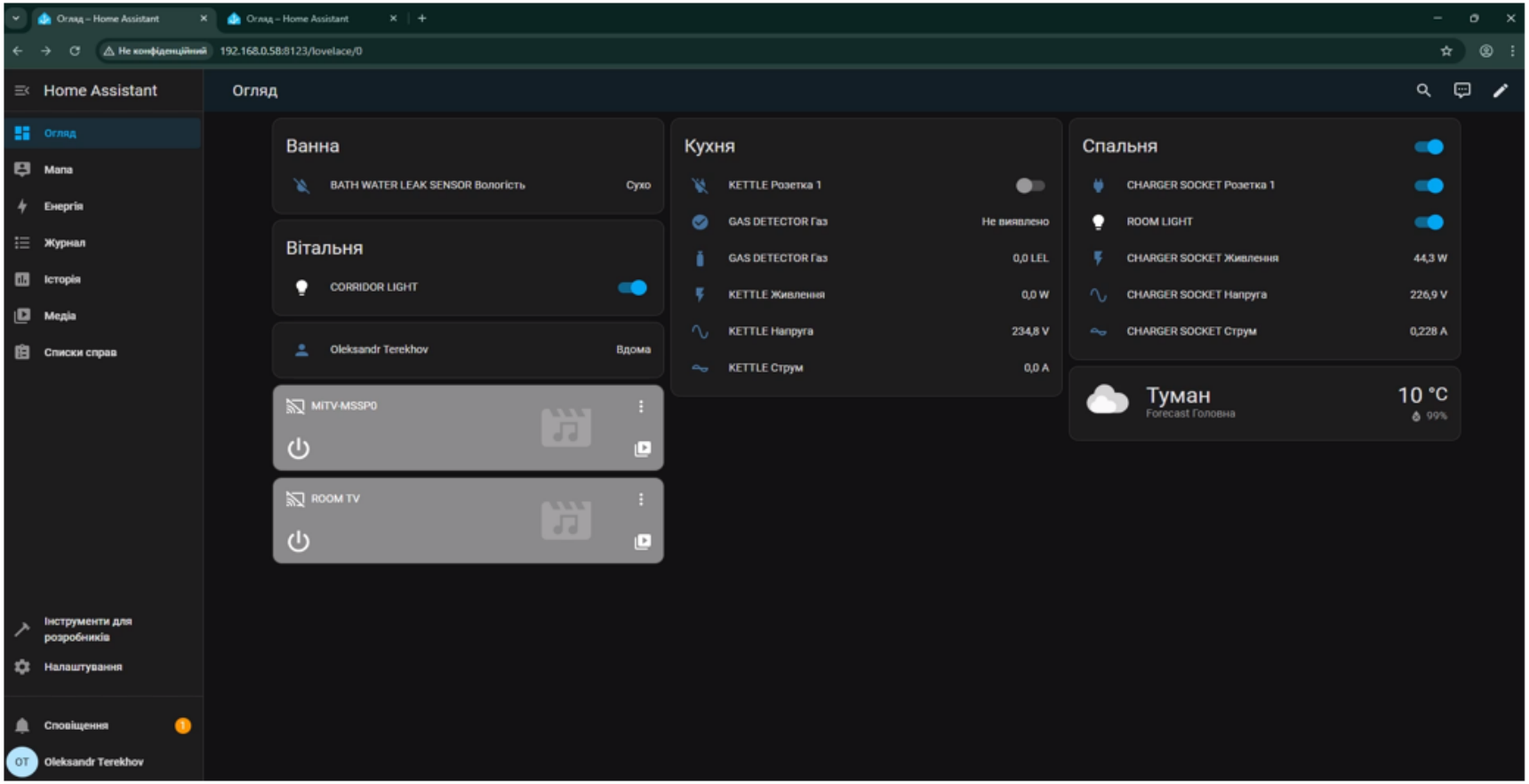

Рис. 2.8 – Автоматично згенерований дашборд всіх сутностів після хмарної інтеграції Tuya

Кастомізований дашборд користувача після локальної інтеграції

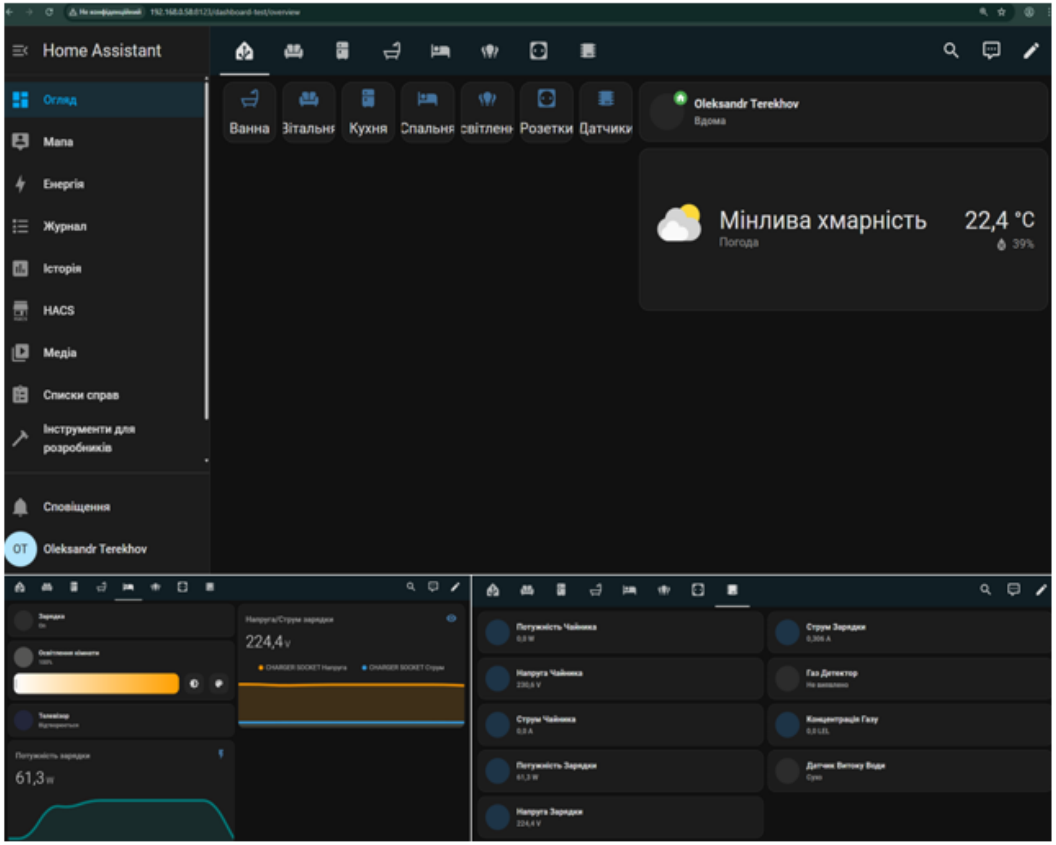

Рис. 2.17 – Власний дашборд всіх сутностей після локальної інтеграції LocalTuya (веб-застосунок)

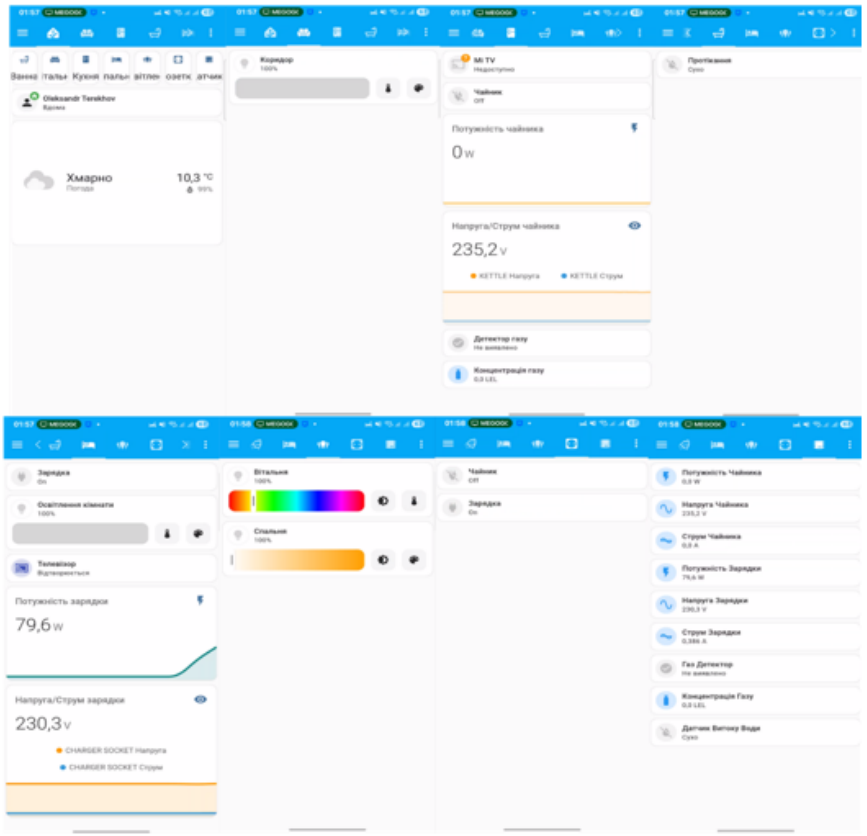

Рис. 2.18 – Власний дашборд всіх сутностей після локальної інтеграції LocalTuya (мобільний застосунок)

Алгоритми комбінованих сценаріїв автоматизації

Рис. 2.22 – Алгоритм автоматизації освітлення у вечірній та нічний час

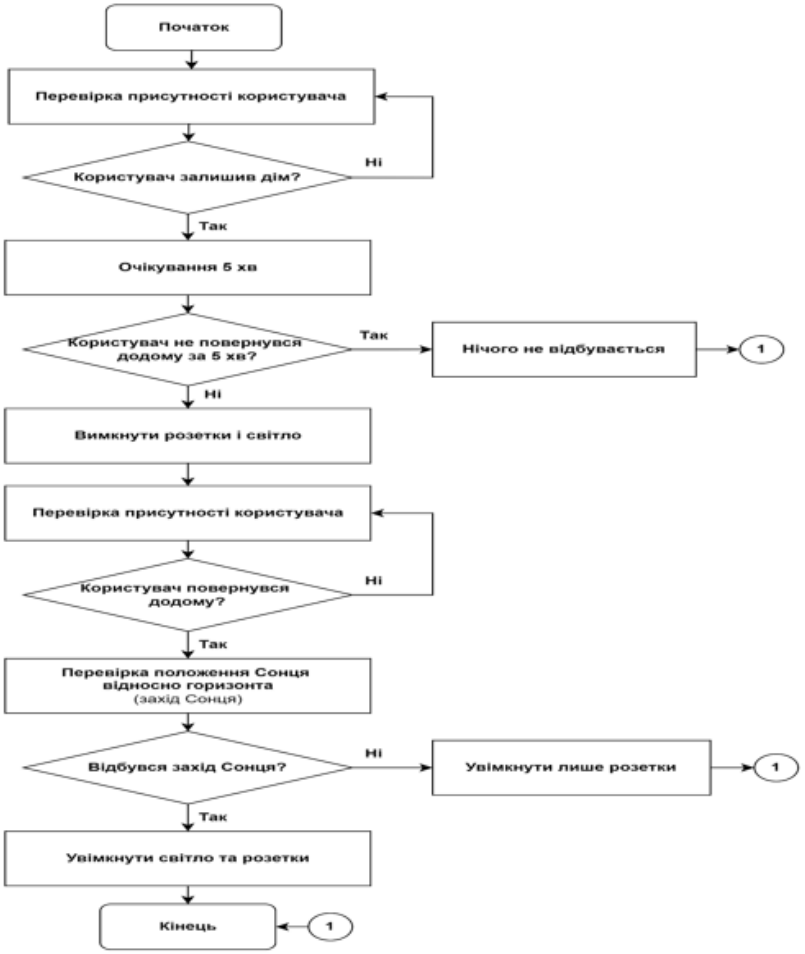

Рис. 2.23 – Алгоритм автоматизації керування при вході/виході з будинку

Аналіз автономної роботи реалізованої системи

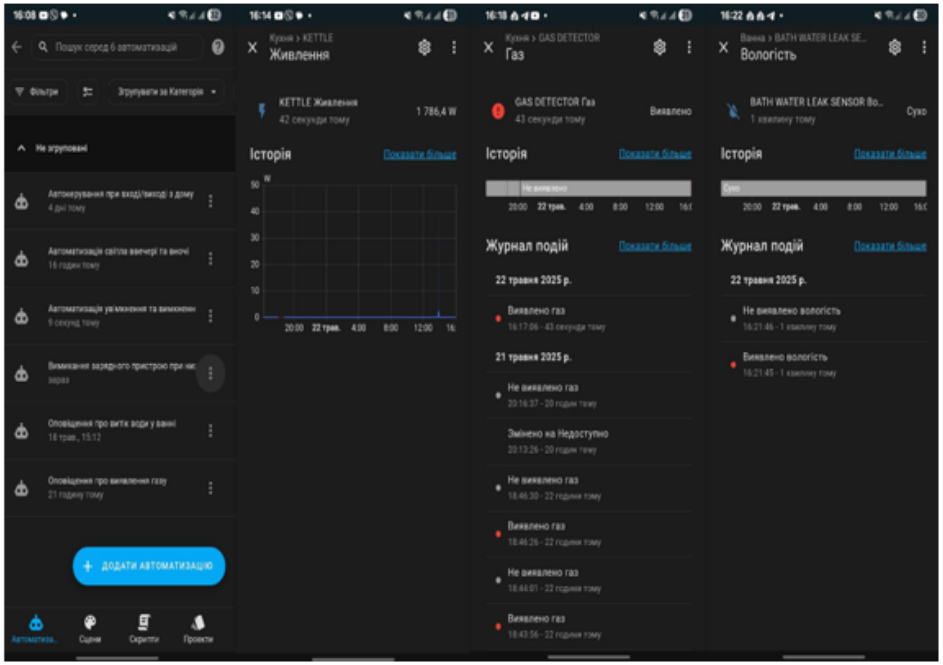

Рис. 2.24 – Робота пристроїв без підключення до Інтернету

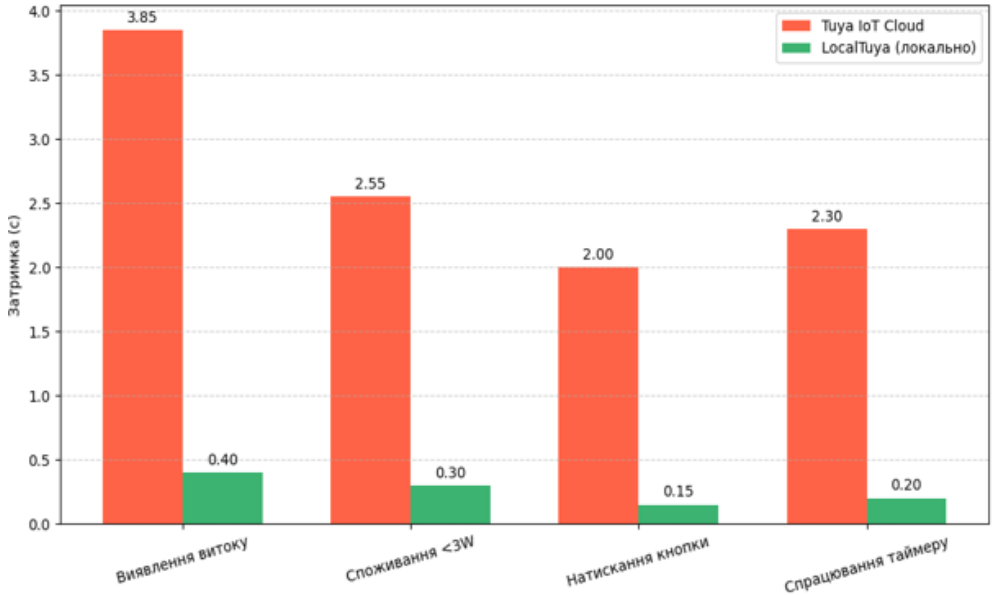

Рис. 2.25 – Вимірювання затримок виконання сценаріїв

Порівняння продуктивності двох систем

Табл. 3.1 – Порівняльна оцінка реалізованої системи з хмарною моделлю

Критерій	**Smart Life (Tuya IoT Cloud)**	**Home Assistant (LocalTuya)**
Швидкодія	Середня (2–4 с)	**Висока (0.2–0.5 с)**
Сценарії	Обмежені	**Гнучкі, багаторівневі**
Автономність	Немає	**Повна**
Кастомізація	Мінімальна	**Повна, включно з UI**
Надійність	Вразлива до збоїв	**Висока**
Логування	Немає / обмежене	**Повне**
Розширюваність	Закрита екосистема	**Відкрита платформа**

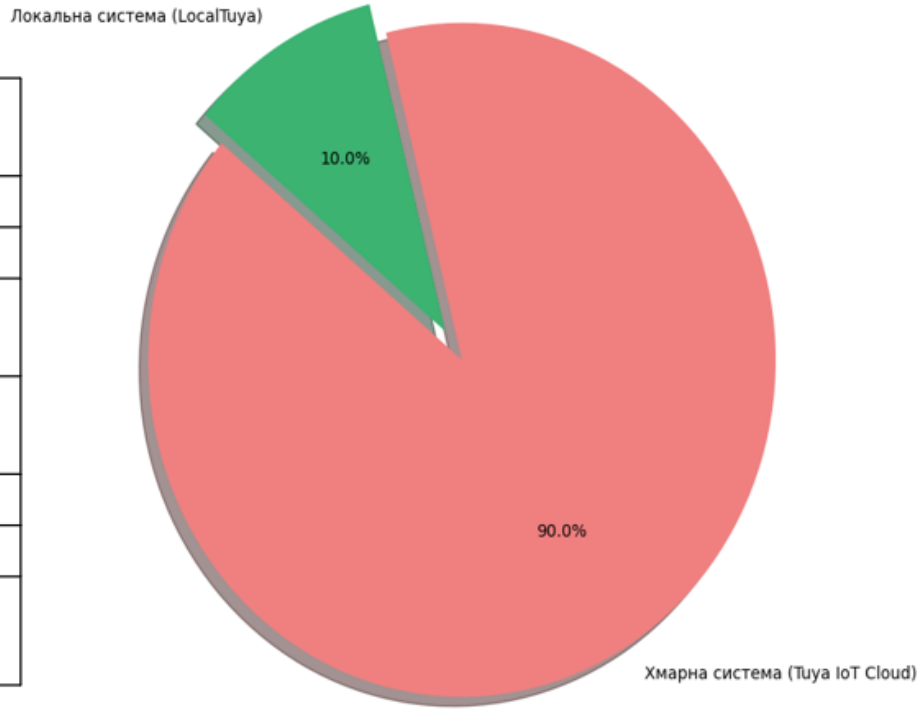

Рис. 3.1 – Діаграма швидкодії хмарної та локальної систем

Кошторис реалізації впровадженої системи

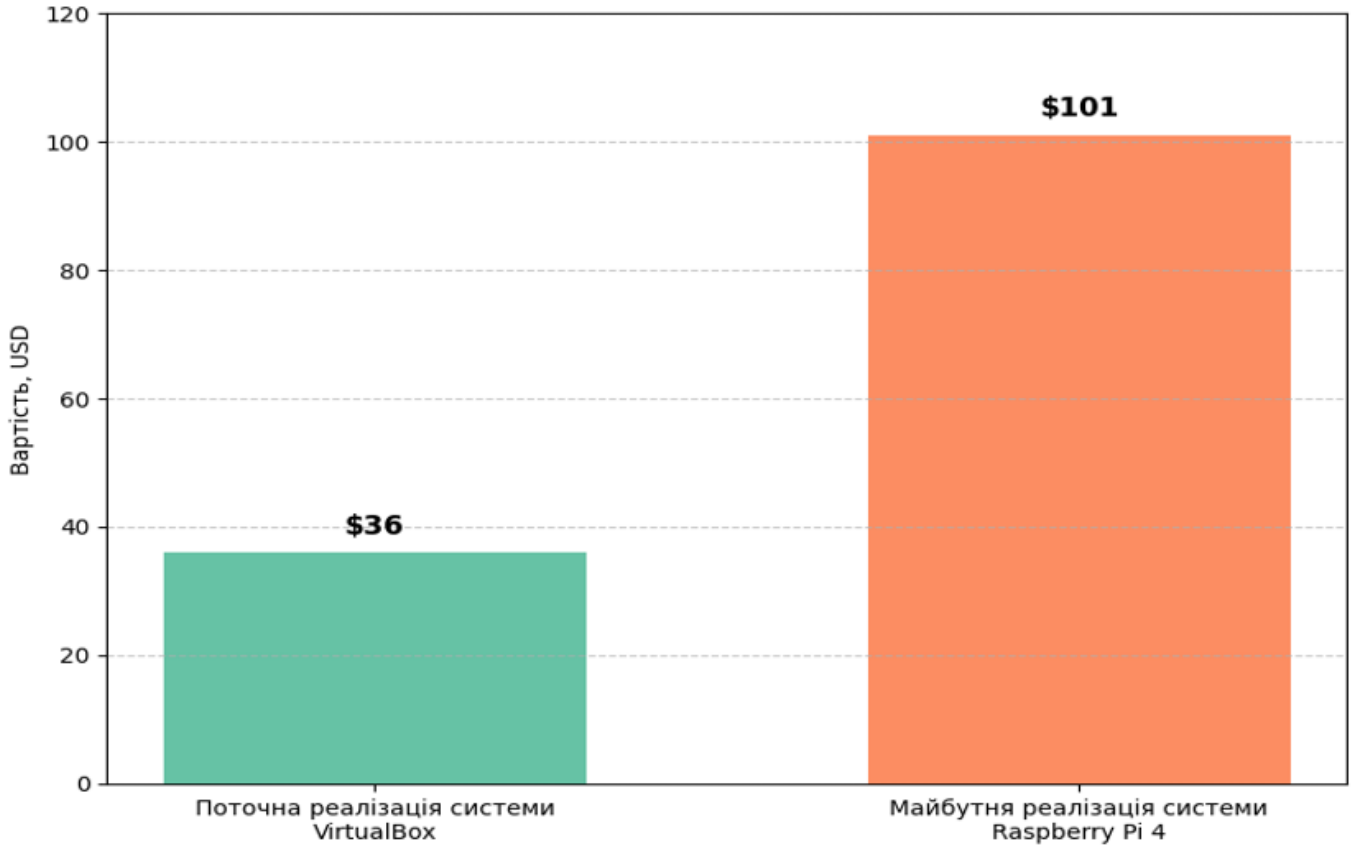

Рис. 3.2 – Діаграма вартості впровадження IoT-системи

Висновки

У ході реалізації було проаналізовано роботу екосистеми Tuya, зокрема мобільної платформи Smart Life, що дозволило оцінити її функціональні можливості, простоту налаштування та типові обмеження. Попри зручний інтерфейс, система виявилася критично залежною від хмарних сервісів і стабільного інтернет-з'єднання, що створює обмеження для автономного використання та ускладнює реалізацію складних сценаріїв автоматизації. Розгортання Home Assistant і налаштування локальної інтеграції через LocalTuya дозволило усунути ці недоліки — пристрої почали працювати повністю автономно, зменшилась затримка реакції, підвищилась стабільність роботи та забезпечено повну незалежність від зовнішніх інфраструктур.

Система, розміщена на локальному сервері, отримала додаткові переваги: можливість зберігати історію подій, здійснювати керування через локальну мережу та оновлювати компоненти з мінімальним ризиком збоїв. За допомогою Home Assistant реалізовано комбіновані сценарії автоматизації з використанням вбудованих умов, тригерів та дій, без потреби у зовнішньому програмному забезпеченні, що дозволило створити ефективне рішення для щоденних завдань, адаптоване до змін конфігурації пристроїв. Таким чином, перехід до Home Assistant забезпечив стабільнішу, безпечнішу та масштабовану систему керування IoT-пристроями, що є оптимальним вибором для побудови сучасного розумного середовища.

Публікації та апробація роботи

1. Терехов О.С., Бондарчук А.П. Програмна реалізація локального керування пристроями Tuya в середовищі Home Assistant. «Застосування програмного забезпечення в ІКТ» : Всеукр. науково-техн. конф., м.Київ, 24 квітня 2025 р. С.535-538.
2. Терехов О.С., Бондарчук А.П. Інтелектуальна модель локального керування об’єктами IoT-мереж у системах автоматизації. «Сучасні інтелектуальні інформаційні технології в науці та освіті» : V Всеукр. науково-практ. конф., м.Київ, 15 травня 2025 р. (подано до публікації).