МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

МЕЛЬНИК Назар Петрович

Розумна система домашньої автоматизації на базі Android з використанням Інтернету речей / Smart home automation system based on Android and using the Internet of Things

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
Н. П. Мельник

Науковий керівник:
к.т.н., доцент, О.Р. Осолінський

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н.М. Васильків

Тернопіль – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
МЕЛЬНИК Назар Петрович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Розумна система домашньої автоматизації на базі Android з використанням Інтернету речей / Smart home automation system based on Android and using the Internet of Things
керівник роботи О.Р. Осолінський, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:

- провести аналіз технологій інтернету речей та розумних систем домашньої автоматизації;
- провести огляд і аналіз існуючих рішень;
- сформувати постановку задачі;
- розробити алгоритмічне та інформаційне забезпечення система домашньої автоматизації на базі Android та IoT;
- розробити архітектуру системи;
- розробити алгоритмічне забезпечення системи;
- розробити інформаційне забезпечення, сформувати інфологічну модель, описати сутності, зв'язки, потоки даних, журнали подій і логіку локального збереження інформації в базі даних SQLite;
- провести вибір та обґрунтування апаратних компонентів та забезпечити їх інтеграцію;
- реалізувати програмно-технологічне забезпечення;
- провести тестування та моделювання роботи системи в симуляційному середовищі та експериментальне дослідження у фізичному середовищі.

5. Перелік графічного матеріалу в роботі:

- архітектура системи розумної системи домашньої автоматизації на базі Android;
- алгоритми функціонування системи;

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент_____ Н. П. Мельник
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ О.Р. Осолінський
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Розумна система домашньої автоматизації на базі Android з використанням Інтернету речей» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 63 сторінок і містить 22 ілюстрації, 4 додатки та 28 використаних джерел.

Метою кваліфікаційної роботи є розробка інтелектуальної системи домашньої автоматизації на базі технологій Інтернету речей (IoT) з використанням мобільного застосунку Android, яка дозволяє керувати побутовими пристроями, здійснювати моніторинг середовища в режимі реального часу, а також автоматизувати виконання дій за заданими сценаріями.

В роботі використано методи структурно-функціонального аналізу, логіко-алгоритмічного моделювання, методи розробки клієнт-серверних систем, REST-архітектуру, а також технології IoT і програмування мікроконтролерів.

В роботі запропоновано архітектуру розумного дому з використанням мікроконтролера ESP32-P4 для збору даних із сенсорів, мікрокомп'ютера Raspberry Pi 4B як центрального хабу, що забезпечує збереження та обробку даних, і Android-застосунку для зручного керування системою. Реалізовано інтерфейс з відображенням температури та динамічними елементами керування освітленням, вентилятором і насосом. Створено симуляційну модель у середовищі Wokwi. Забезпечено локальне збереження даних у базі SQLite, без використання хмарних сервісів. Передбачено адаптивний інтерфейс для мобільних пристроїв із підтримкою push-повідомлень.

Практична цінність: реалізована система є масштабованою, енергоефективною, безпечною для використання в умовах реального житла.

Ключові слова: ІНТЕРНЕТ РЕЧЕЙ, РОЗУМНИЙ ДІМ, ESP32, Raspberry Pi, МОБІЛЬНИЙ ЗАСТОСУНОК, Android, АВТОМАТИЗАЦІЯ, СЕНСОРИ, РЕЛЕ, Flask, SQLite.

ANNOTATION

Qualification thesis titled “Smart home automation system based on Android and using the Internet of Things” for obtaining the bachelor's degree in the specialty 122 “Computer Science” of the educational program “Computer Science” consists of 63 pages and includes 22 illustrations, 4 appendices, and 28 references.

The purpose of this qualification work is to develop an intelligent home automation system based on Internet of Things (IoT) technologies using an Android mobile application, enabling users to control household devices, monitor the environment in real time, and automate actions according to predefined scenarios.

The work applies methods of structural-functional analysis, logical-algorithmic modeling, client-server system development methods, REST architecture, as well as IoT technologies and microcontroller programming.

The proposed smart home architecture includes the ESP32-P4 microcontroller for sensor data collection, a Raspberry Pi 4B microcomputer as the central hub for data storage and processing, and an Android application for convenient system control.

An interface has been implemented to display temperature data and dynamically control lighting, a fan, and a pump. A simulation model was created in the Wokwi environment. Local data storage is provided via an SQLite database, without the use of cloud services. The mobile interface is adaptive and supports push notifications.

Practical value: the implemented system is scalable, energy-efficient, and safe for use in real residential environments.

Keywords: INTERNET OF THINGS, SMART HOME, ESP32, Raspberry Pi, MOBILE APPLICATION, ANDROID, AUTOMATION, SENSORS, RELAY, Flask, SQLite.

ЗМІСТ

ВСТУП

В сучасному світі зростає потреба у створенні комфортного, безпечного та енергоефективного житлового простору, що спонукає до активного розвитку розумних систем автоматизації будинку. Ідея автоматизації домашнього середовища за допомогою новітніх технологій є не новою, однак саме поява Інтернету речей (IoT) значно трансформувала підходи до реалізації таких систем. Смарт-технології проникають у повсякденне життя з метою підвищення зручності користування побутовими пристроями, забезпечення енергоефективності, поліпшення якості життя та підвищення рівня безпеки.

Початково автоматизація будинку асоціювалася переважно з освітленням та управлінням температурним режимом, однак із розвитком бездротових технологій, таких як Wi-Fi, ZigBee, Bluetooth, GSM, і поширенням недорогих мікроконтролерів на кшталт Arduino чи Raspberry Pi, спектр можливостей значно розширився. Зокрема, реалізуються сценарії автоматичного керування освітленням, кліматом, безпекою, поливом, побутовою технікою, медичним моніторингом і навіть розважальними системами.

В літературних джерелах, що аналізувалися, простежується поступова еволюція підходів до побудови домашньої автоматизації: від простих Bluetooth-з'єднань з ручним управлінням до повністю автономних систем, що взаємодіють з хмарними сервісами, застосовують методи машинного навчання, здатні адаптуватися до поведінки користувача та вивчати його звички.

Також важливо зазначити, що домашня автоматизація може відігравати ключову роль у підтримці людей з особливими потребами, осіб похилого віку та людей з обмеженою мобільністю. Інтелектуальні системи можуть допомогти цим користувачам самостійно контролювати освітлення, температуру, двері чи вікна, створюючи безпечне і зручне середовище.

Попри всі переваги, існують деякі проблеми. Основні з них — це безпека та захист персональних даних. Будь-яка IoT-система є потенційною мішенню для кіберзагроз. Відсутність єдиних стандартів, фрагментованість ринку, обмеження у

пропускній здатності мереж, а також потреба в енергоефективності — все це вимагає комплексного підходу до проектування архітектури таких систем.

Метою кваліфікаційної роботи є розробка розумної системи домашньої автоматизації на базі Android з використанням Інтернету речей

Завдання кваліфікаційної роботи полягають у послідовному виконанні наступних етапів:

– провести аналіз технологій інтернету речей та розумних систем домашньої автоматизації;

– провести огляд і аналіз існуючих рішень;

– сформувати постановку задачі;

– розробити алгоритмічне та інформаційне забезпечення система домашньої автоматизації на базі Android та IoT;

– розробити архітектуру системи;

– розробити алгоритмічне забезпечення системи;

– розробити інформаційне забезпечення, сформувати інфологічну модель, описати сутності, зв'язки, потоки даних, журнали подій і логіку локального збереження інформації в базі даних SQLite;

– провести вибір та обґрунтування апаратних компонентів та забезпечити їх інтеграцію;

– реалізувати програмно-технологічне забезпечення;

– провести тестування та моделювання роботи системи в симуляційному середовищі та експериментальне дослідження у фізичному середовищі.

Об’єкт дослідження - інформаційні процеси моніторингу, керування і автоматизації фізичних об'єктів у середовищі розумного дому на базі технологій Інтернету речей (IoT).

Предмет дослідження - методи проєктування, реалізації та тестування розумних систем домашньої автоматизації.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах студентської науково-практичної конференції “Інтелектуальні інформаційні

технології в прикладних дослідженнях” (IITAR – 2025), м. Тернопіль, Україна, 27-29 травня 2025 р.

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ТЕХНОЛОГІЙ ІНТЕРНЕТУ РЕЧЕЙ ТА РОЗУМНИХ СИСТЕМ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ

1.1 Домашня автоматизація та IoT

В сучасному світі, де цифрові технології щодня проникають у нові сфери життя, усе більше уваги приділяється створенню комфортного та безпечного середовища в оселі. Серед найактуальніших напрямів технологічного розвитку опинилися інтелектуальні системи домашньої автоматизації, що базуються на технологіях Інтернету речей. Ці системи дозволяють здійснювати моніторинг і керування домашніми пристроями, використовуючи мобільні застосунки, зокрема на платформі Android. Такий підхід значно полегшує повсякденне життя, робить його зручнішим, а також сприяє ефективнішому використанню енергетичних ресурсів.

Розумна система домашньої автоматизації — це інтеграція різних побутових приладів, сенсорів і контролерів у єдине інформаційне середовище, яке реагує на зміни навколишнього середовища та виконує дії відповідно до заданих сценаріїв або команд користувача. Завдяки швидкому розвитку бездротових технологій і доступних апаратних платформ, таких як Arduino, Raspberry Pi чи ESP8266, стало можливим створення недорогих і гнучких рішень для керування будинком.

Зазвичай такі системи включають у себе набір сенсорів, які зчитують параметри навколишнього середовища, наприклад температуру, вологість, рівень освітлення, наявність руху або диму. Ці дані передаються до центрального контролера, що обробляє інформацію та приймає рішення про увімкнення чи вимкнення певних пристроїв. Наприклад, система може автоматично увімкнути світло при виявленні руху в кімнаті або вимкнути обігрівач, коли досягнуто задану температуру. Комунікація між пристроями здійснюється або через дротову мережу, або бездротовими протоколами, такими як Wi-Fi, Bluetooth, ZigBee або інші.

Однією з основних переваг використання платформи Android у системах домашньої автоматизації є її доступність та поширеність. Сьогодні практично кожен користується смартфоном на Android, а отже, не потрібно додаткових витрат

на обладнання для керування системою. Android-додатки дозволяють реалізувати інтуїтивно зрозумілий інтерфейс, через який користувач може переглядати параметри системи, отримувати повідомлення про події або змінювати налаштування. Це спрощує процес адаптації користувача до системи та розширює її функціональні можливості.

З технічного боку, апаратна частина таких систем зазвичай базується на доступних мікроконтролерах. Наприклад, платформа Arduino є простою у використанні та має широку спільноту, що розробляє відкриті бібліотеки для роботи з різними сенсорами. Вона підходить для створення простих рішень, де потрібно зчитувати показники сенсорів і керувати реле для вмикання або вимикання приладів [2]. Raspberry Pi, на відміну від Arduino, є повноцінним одноплатним комп'ютером, який може запускати операційну систему, працювати як сервер і обробляти складніші задачі. У випадках, коли потрібна більша обчислювальна потужність або можливість зберігати історію подій, Raspberry Pi стає ідеальним рішенням. Інший популярний варіант — це модулі NodeMCU на базі ESP8266, які мають вбудований Wi-Fi та дозволяють створювати компактні й енергоефективні системи для віддаленого контролю [6].

Кожна система автоматизації має свою архітектуру, яка адаптується під конкретні завдання. Наприклад, у багатьох випадках застосовується підхід, коли вся система будується навколо центрального контролера, який обробляє всі дані та приймає рішення. Такий контролер отримує інформацію від сенсорів, аналізує її та керує виконавчими елементами — реле, двигунами, лампами, вентиляторами. Користувач у цьому випадку взаємодіє із системою через Android-додаток, який надсилає команди на контролер або отримує з нього дані про стан пристроїв [1; 3]. У деяких випадках використовуються розподілені системи, де частина логіки виконується безпосередньо на периферійних пристроях, що дозволяє підвищити надійність та швидкодію [6].

Особливістю сучасних систем є також підтримка віддаленого доступу. Користувачі можуть перебувати далеко від дому й одночасно мати змогу перевірити, чи вимкнене світло, чи зачинені двері, або ж змінити налаштування

кліматичної системи перед поверненням додому. Це досягається завдяки використанню інтернет-з'єднання, хмарних сервісів або створенню локального сервера на базі Raspberry Pi [4, 5].

Функціональні можливості таких систем дуже широкі. Найпоширеніші сценарії включають автоматичне регулювання освітлення, контроль температури, виявлення присутності людей, а також інтеграцію з системами відеоспостереження. Наприклад, за допомогою сенсорів руху система може вмикати освітлення в коридорі, коли хтось проходить повз. Або, використовуючи датчик вологості ґрунту, система може активувати полив рослин у теплиці. Іншим прикладом є автоматичне відкривання штор вранці чи закривання їх при підвищеній температурі зовні [3, 6].

Попри численні переваги, такі системи мають і певні обмеження. Зокрема, одним із викликів є безпека даних. Оскільки система підключена до інтернету, важливо забезпечити захист від несанкціонованого доступу. Це передбачає використання шифрування даних, автентифікації користувачів, регулярного оновлення програмного забезпечення. Ще одним викликом є стабільність мережі. Якщо з'єднання з інтернетом переривається, частина функціональності може бути недоступною, тому деякі системи передбачають резервний режим або автономну роботу за певних умов [6].

Загалом, впровадження розумних систем автоматизації на базі IoT та Android відкриває нові можливості для підвищення якості життя. Вони сприяють зниженню витрат на енергію, підвищенню рівня безпеки, зручності користування домашніми приладами. Дослідження й публікації останніх років демонструють, що такі системи успішно реалізуються в різних країнах, при цьому використовуються відкриті платформи, які дозволяють адаптувати проєкти під конкретні умови. Наприклад, система qToggle, описана в роботі 2021 року, демонструє, як можна об'єднати сенсори на базі ESP8266 у єдину мережу з відкритим API, що робить її легкою для розширення та налаштування [6]. Інші приклади реалізацій демонструють використання Raspberry Pi як локального сервера, до якого

підключаються сенсори, реле та Android-додатки, створені спеціально для керування цією системою [4; 5].

Таким чином, розумна система домашньої автоматизації на базі Android і технологій Інтернету речей — це перспективний напрямок, який не лише покращує умови проживання, а й створює нові можливості для досліджень, інженерної творчості та інновацій. У майбутньому такі системи будуть розвиватися у напрямку глибшої інтеграції зі штучним інтелектом, самонавчанням і персоналізацією, забезпечуючи ще вищий рівень комфорту, безпеки та енергоефективності.

1.2 Огляд і аналіз існуючих рішень

Останнім часом Інтернет речей (IoT) захоплює весь світ. IoT — це система взаємопов’язаних пристроїв, які можна контролювати та відстежувати дистанційно через мережу Інтернет. За останні роки ця концепція зазнала інтенсивного розвитку [1]. Зараз він використовується в різних секторах, включаючи розумні будинки, хмарні обчислення (виявлення потоків) [2] і охорону здоров’я (біомедичні захворювання) [3], телемедицину, промислові установки та інші програми.

Вбудовані в IoT технології бездротових сенсорних мереж дозволяють підключати інтелектуальні пристрої по всьому світу з розширеною функціональністю. Безпровідна система автоматизації в межах розумного дому являє собою сукупність сенсорів та виконавчих пристроїв, здатних до взаємодії між собою та спільного використання ресурсів. Такий підхід є ключовим елементом у побудові інтелектуальних житлових середовищ. Концепція «розумного дому» є складовою частиною парадигми Інтернету речей (IoT) та спрямована на впровадження автоматизованих рішень у побуті [4]. Це значний прогрес, який дозволяє споживачам віддалено контролювати та контролювати побутову техніку, підключаючи її до Інтернету.

Доступно декілька інтелектуальних пристроїв, у тому числі вимикачі світла, якими можна керувати за допомогою голосової команди смартфона [5]. Розумна система зрошення [6] і термостати, які можуть змінювати температуру всередині

при створенні звітів про енергоспоживання, зменшенні витрат води тощо. Протягом останніх кількох років рішення для розумного будинку були більш популярними.

На рисунку 1.1 представлена система автоматизації розумного дому з використанням кількох підключених до Інтернету речей пристроїв.

Рисунок 1.1 – Загальна схема інтелектуальної системи домашньої автоматизації на основі IoT із використанням різних сенсорних пристроїв.

Системи домашньої автоматизації - це технічний спосіб контролю, зворотного зв'язку, дії побутової техніки та інтелектуального моніторингу на основі запитів мешканців будинку. Останнім часом більшість перемикань виконується вручну та не використовує концепцію IoT. Пристрій керування дуже корисний для користувача, оскільки він дозволяє керувати всіма пристроями, підключеними до системи, з одного місця. Побутовою технікою, такою як вентилятори, освітлення та вимикачі, можна керувати дистанційно через центральну панель керування [7].

Однією з найважливіших переваг систем домашньої автоматизації є легкий контроль і управління різними пристроями, такими як настільні комп’ютери, ноутбуки, смартфони, розумні годинники тощо. Переваги системи розумної домашньої автоматизації полягають у простоті керування й керування всіма побутовими приладами, включаючи керування освітленням, регулювання

температури, керування телевізором та кондиціонером, з метою безпеки використання відеокамери тощо. Що економить багато часу та грошей.

Протягом останніх двох десятиліть різні дослідники пропонували систему домашньої автоматизації на основі IoT. У системах домашньої автоматизації реалізовано різні технології; кожна має переваги та недоліки. Наприклад, система домашньої автоматизації на основі Bluetooth [8] є недорогою, швидкою та простою в установці, але вона може спілкуватися лише на короткій відстані. ZigBee та GSM (Глобальна система мобільного зв'язку) — це бездротові технології, які працюють для зв'язку на великій відстані. Zigbee [9] — це стандарт бездротової сітчастої мережі для пристроїв із живленням від батареї в додатках бездротового керування та моніторингу. Він розроблений таким чином, щоб мати низьку вартість і низьке енергоспоживання. Крім того, він має низьку швидкість передачі даних, низьку швидкість передачі та низьку стабільність мережі.

Крім того, він має високу вартість обслуговування. Технологія на основі Wi-Fi реалізована в літературі [10]. Технологія Wi-Fi має кілька переваг перед ZigBee і Z-Wave, включаючи вартість, складність (яку можна інтерпретувати як простоту) і доступність.

Крім того, розумні пристрої, які підтримують Wi-Fi, зазвичай доступні. На додаток Wi-Fi є дешевшим варіантом, за допомогою якого пристрої легко підключаються. У сучасних умовах, коли Wi-Fi є базовою складовою більшості домогосподарств, значно зручніше використовувати пристрої з вбудованою підтримкою бездротового зв'язку. Така технологія забезпечує легке підключення та використання, що дозволяє користувачу з мінімальними зусиллями інтегрувати необхідну кількість пристроїв для налаштування системи домашньої автоматизації. Це робить Wi-Fi ідеальним для домашнього використання, тому що немає додаткового обладнання для підключення до мережі. Щоб реалізувати інтелектуальний HAS, користувачеві потрібна тільки базова конфігурація.

Крім того, Wi-Fi не призначений для створення сітчастої мережі. Він споживає в шість разів більше енергії, ніж аналогічні пристрої, які використовують Z-Wave, ZigBee або Bluetooth, тому що більшість маршрутизаторів Wi-Fi можуть

підключати від двадцяти до двадцяти п’яти пристроїв одночасно. Крім того, Wi-Fi пропонує численні переваги порівняно з Ethernet, наприклад просте підключення та доступність різних пристроїв, можливість розширення (що дозволяє додавати нові пристрої без додаткового кабелю), дешевшу вартість і необхідність наявності лише однієї точки доступу. Але виникають труднощі зі швидкістю з’єднання (високошвидкісна мережа Wi-Fi значно повільніша за швидкість дротової мережі), а також хвилювання щодо безпеки та конфіденційності в Інтернеті. Для цілей апаратних компонентів домашньої автоматизації з відкритим вихідним кодом і недорогих, таких як плати Raspberry Pi і мікроконтролерів Arduino (MCU), а також різні датчики які використовуються.

У [11] було досліджено багато методів домашньої автоматизації за допомогою плат Arduino. Arduino — це апаратна платформа з відкритим вихідним кодом, проста у використанні, недорога, дуже гнучка та проста в програмуванні [12].

Крім того, велике суспільство є значною перевагою цього продукту. З іншого боку, Arduino не була призначена для роботи зі значною складністю більш складних проектів. Raspberry Pi (RPi) має надійне рішення для складних програм, які потребують обробки в реальному часі. RPi — це цікава технологія, доступна за ціною, значно нижчою, ніж будь-який мобільний пристрій [13].

Більшість проектів реалізовували відкритий вихідний код Raspberry Pi для онлайн-користувачів. Ці спільноти завжди в захваті від нових проектів, які розробляються.

Python є кращою мовою для розробки програмного забезпечення для Raspberry Pi, оскільки порівняно з іншими мовами програмування вона простіша у використанні (вимагає менше рядків коду та має нижчий рівень складності). Raspberry Pi недорогий, має низький вплив на навколишнє середовище та не потребує системи охолодження. В роботі [14] представлено ідеї використання Raspberry Pi у поєднанні з інтелектуальною домашньою автоматизацією.

Недорогі модулі Wi-Fi, відомі як чіпи ESP8266, є ідеальним вибором для використання в проектах, пов’язаних з Інтернетом речей (IoT). ESP8266 має єдине

процесорне ядро, яке працює на частоті 80 мегагерц. Приклади проектів домашньої автоматизації представлено в роботах [15], в яких використовувалися чіпи ESP8266.

В таблиці 1.1 представлено тенденції та характеристики розумного HAS, наявні в попередніх літературах джерелах.

Таблиця 1.1 – Характеристики розумного HAS

Контролер	Інтерфейс користувача	Зв'язок	Застосування
PIC (Microchip Technology, Chandler)	мобільний застосунок	Bluetooth	Керування роботою внутрішніх пристроїв
Arduino	мобільний застосунок	Bluetooth	Керування внутрішніми й зовнішніми пристроями на короткій відстані
Arduino Mega	мобільний застосунок	Ethernet, ZigBee	Керування внутрішніми пристроями
Raspberry Pi	мобільний застосунок	ZigBee, Wi-Fi	Регулювання вологості, температури, освітлення, руху та електричних струмів
Сервер / ноутбук	мобільний застосунок	ZigBee	Керування внутрішніми пристроями (не реалізовано на практиці)
TI-CC3200	мобільний застосунок	Wi-Fi	Керування внутрішніми пристроями та моніторинг вологості ґрунту
Raspberry Pi, Arduino Mega	Веб та мобільний застосунок	Wi-Fi	Автоматизація житла (внутрішня й зовнішня), керування безпекою, прогнозування енергоспоживання, керування живленням і сумісність з Google Assistant

Інші системи домашньої автоматизації мають комерційні інструменти або платформи, такі як Dom intell, Qivicon, HomeSeer, Loxone.

Цей тип системи відноситься до категорії «комерційних платформ». Вони пропонують широкий вибір пристроїв домашньої автоматизації від кількох різних виробників, а також численні засоби автоматизації, включаючи систему замків,

контроль температури, систему освітлення, систему навколишнього середовища, HomeSeer, відеоспостереження Qivicon та функції захисту від вторгнення. Кожне рішення містить мобільний додаток, який можна використовувати для керування системами. Що стосується вартості, все залежить від квадратних метрів будинку та вимог користувача. За оцінками, мінімальна вартість коштуватиме від 2000 до 2200 доларів [16].

Існує велика кількість систем домашньої автоматизації з відкритим вихідним кодом [17]. Серед них найбільш потужними та розвиненими представниками спільноти є Home Assistant [18] і OpenHAB [19]. Обидва рішення мають схожі цілі та підтримують інтеграцію великої кількості пристроїв. Водночас, інтеграція пристроїв в OpenHAB вимагає знань щодо форматів команд і є досить складною та трудомісткою. У свою чергу, Home Assistant відзначається більшою простотою використання, однак потребує значних зусиль на етапі первинного налаштування.

Мобільні застосунки, пов’язані з такими системами, зазвичай є менш гнучкими і можуть видаватися складними для новачків. Як показано в дослідженні [20], такі рішення містять достатню функціональність, більшість конфігурацій здійснюється через вебінтерфейс, а додаткові можливості забезпечуються через плагіни. Проте інтерфейс користувача не завжди є інтуїтивно зрозумілим, що дослідники вважають одним з обмежень подібних підходів.

Також потрібно відзначити дві французькі компанії — Calaos [21] та Jeedom [22], які активно беруть участь у розробці відкритих систем автоматизації будинку. Однак значна частина документації, форумів і підтримки доступна лише французькою мовою, що створює мовний бар’єр для ширшого міжнародного використання.

Порівняльний аналіз ключових рішень домашньої автоматизації з відкритим кодом представлено у таблиці 1.2, де розглядаються їх основні функціональні можливості.

Таблиця 1.2 – Системи автоматизації за функціональністю

Система	API	Мова програмування	Інші можливості
Open Hub	Представницька передача стану (REST)	Java	Розширена документація, багато протоколів, веб-інтерфейс, MQTT (черга повідомлень)
Domoticz	На основі JSON (JavaScript Object Notation)	C++	Невелика кількість протоколів, MQTT, веб-інтерфейс, розширена документація
Jeedom	На основі HTTP і JSON	PHP	Розширена документація, багато плагінів, веб-інтерфейс, багато протоколів
Home Assistant	Python / REST	Python	Розширена документація, багато протоколів, веб-інтерфейс, MQTT

Платформи відрізняються між собою за рядом ознак, зокрема за мовою програмування, що використовується для розробки, набором API, кількістю доступних плагінів і підтримуваних протоколів, а також обсягом і різноманітністю супровідної документації.

Автори статті [23] надають комплексний аналіз п’ятнадцяти різних платформ з відкритим кодом. Метою цього дослідження є представлення HAS; Інша система була розроблена для багаторазової автоматизації будівель/будинків, таких як безпека та контроль доступу [24], контроль приладів, включаючи термостати, освітлення, змінний струм тощо), керування енергією та живленням [25].

Це дуже корисно в таких сферах сталого розвитку, як екологічна стійкість (технології управління енергією) [26], економічна стійкість (рентабельні технології проектування розумного будинку) [27], стійкість від системи автоматизації розумних будинків до розумного простору [28]. Ідеї, представлені в цьому дослідженні, є розширеною версією дослідження [29]. Автоматична система будівництва була запропонована в цьому дослідженні, щоб мінімізувати розповсюдження та розповсюдження COVID-19 під час пандемії на робочому

місці, не даючи людям торкатися певних поверхонь і предметів. Крім того, це рішення було покликане допомогти будинкам, які перебувають в аварійному стані. Основна мета цього дослідження — зосередитися на розумній побутовій техніці, а не на ситуації з COVID-19.

У багатьох дослідницьких статтях представлені різні технології зв'язку, такі як використання бездротової або реалізованої технології бездротового зв'язку для надсилання даних із вузлів на пристрої зберігання даних, з'єднання датчика з вузлами тощо. Налаштування HAS, Wi-Fi або Ethernet достатньо. Більшість пристроїв IoT підтримують Wi-Fi. Вибір найкращої технології, як-от процесора (контролера) для системи домашньої автоматизації, залежить від потреб користувача. Більшість літератури вважали за краще використовувати сімейство Raspberry через його сильні обчислювальні можливості, які роблять потужнішими, ніж плати Arduino.

Керування програмованими пристроями за допомогою стека протоколу керування передачею/протоколу Інтернету (TCP/IP) за допомогою простих запитів HTTP — це концепція, що лежить в основі одноплатних комп'ютерів HAS, а мікроконтролери з підтримкою TCP/IP — два приклади типів систем, які підпадають під цю категорію. Основна мета HAS полягала в тому, щоб висунути стандарт, який би дозволив керувати різними пристроями, розгортати їх і спілкуватися з ними. HAS використовує технології, які вже існують і широко впроваджуються, включаючи RESTful API, створені на основі HTTP, які передають дані у форматі JSON. Нижче наведено нові функції HAS special:

1. Єдине уніфіковане рішення, яке поєднує в собі всі інтегровані функції;

2. Керування пристроєм і забезпечення.

3. Спеціальна мікропрограма оновила всі пристрої за допомогою того самого унікального API.

4. Між приводами та датчиками всередині мережі реалізуються складні та інтелектуальні правила.

5. Архітектура головний-підлеглий, яка є ієрархічною та забезпечує масштабованість і гнучкість.

6. Немає необхідності підключатися до хмари (з міркувань конфіденційності та безпеки, тому дані користувача не знаходяться в межах локальної мережі).

7. Інтегрована програма для смартфонів із плагіном найкраще працює на всіх найпопулярніших платформах, настільних чи мобільних, включаючи Android, Windows, iOS, macOS, Linux тощо.

1.3 Постановка задачі

В сучасному світі розумні системи автоматизації житла на базі Інтернету речей (IoT) стають невіддільною частиною побуту, сприяючи покращенню рівня комфорту, безпеки та енергоефективності. Проте, попри активний розвиток галузі, більшість існуючих рішень є або комерційно складними для масового користувача, або вимагають значних технічних знань для розгортання та налаштування. Водночас, існує потреба в створенні інтегрованої, доступної та адаптивної системи, яка дозволить здійснювати інтелектуальний контроль над побутовими приладами з мобільного пристрою з використанням Wi-Fi, забезпечуючи просту інтеграцію з існуючими платформами Android, мінімізуючи при цьому затрати на розробку та обслуговування.

Актуальність проблеми полягає також у необхідності забезпечення надійної комунікації між пристроями, локального збереження даних, захисту персональної інформації користувачів та відсутності залежності від сторонніх хмарних сервісів. Система повинна мати можливість масштабування, підтримку декількох ролей доступу, а також здатність до адаптації на основі заданих сценаріїв або поведінкових моделей користувача.

Метою є розробка архітектури інтелектуальної системи домашньої автоматизації, яка базується на пристроях IoT (зокрема, ESP8266, Raspberry Pi), використовує мобільний Android-додаток для керування, підтримує локальне зберігання даних та забезпечує енергоефективне управління побутовими приладами.

Для досягнення цієї мети необхідно вирішити такі завдання:

1. Розробити алгоритмічне та інформаційне забезпечення система домашньої автоматизації на базі Android та IoT.

2. Розробити архітектуру системи.

3. Розробити алгоритмічне забезпечення системи.

4. Розробити інформаційне забезпечення, сформувати інфологічну модель, описати сутності, зв'язки, потоки даних, журнали подій і логіку локального збереження інформації в базі даних SQLite.

5. Провести вибір та обґрунтування апаратних компонентів та забезпечити їх інтеграцію.

6. Реалізувати програмно-технологічне забезпечення.

7. Провести тестування та моделювання роботи системи в симуляційному середовищі та експериментальне дослідження у фізичному середовищі.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ НА БАЗІ ANDROID ТА IOT

2.1 Розробка архітектури системи

Для функціональної конфігурації HAS зазвичай достатньо традиційної мережі Wi-Fi та Ethernet. Система використовує різноманітні апаратні компоненти, включаючи плати Raspberry Pi 3, ESP 8266 і модулі Wi-Fi.

Модель Raspberry Pi 4 була обрана завдяки прогресу, досягнутому в порівнянні з попередніми моделями. Bluetooth тощо недоступний на RPi1 і 2. На RPi помітною функцією є ряд доступних контактів введення/виведення загального призначення (GPIO) [37,38].

Усі поточні плати Raspberry Pi мають 40-контактний роз'єм GPIO [30]. Плата RPi може виконувати три функції в установці HAS:

– вона може служити основним пристроєм HAS з платами реле та датчиками;

– може бути головним центром для інших підключених пристроїв для запуску Tuya (платформа китайських інтелектуальних пристроїв, яка надає хмарні служби для пристроїв на основі ESP8266/ESP8285).

Деякі пристрої можуть інсталювати програму Convert (OC) ESP, інструмент, який замінює пропрієтарну мікропрограму Tuya на саморобну без видалення пристрою. Важливо відзначити, що він сумісний лише з пристроями на базі Tuya.

Мікроконтролер є важливим компонентом HAS на основі IoT. З точки зору продуктивності, модуль Wi-Fi ESP 8266 містить набір високоінтегрованих бездротових систем на чіпі (SoC), які забезпечують інтегроване та незалежне мережеве рішення Wi-Fi. Версія мікросхеми ESP8266EX є однією з найінтегрованіших чіпів Wi-Fi, доступних на ринку. Крім того, ESP8266EX містить удосконалену версію 32-розрядного ЦП серії L106 Diamond від Tensilica, а також вбудовану SRAM. На додаток до сімнадцяти контактів GPIO, які можна призначити для різних цілей шляхом встановлення належних регістрів, ESP8266EX має два

контакти живлення, контакт скидання, один провід заземлення та два висновки годинника.

У більшості випадків використовуються датчики і виконавчі механізми пристроїв з підключенням вгору. Дотримання версії прошивки на пристроях є, мабуть, однією з важливих справ, але її часто забувають при роботі з багатьма пристроями. Щоб зробити цю роботу більш доступною, HAS дозволяє оновити мікропрограму пристроїв різних видів і моделей простим способом.

HAS API — це простий HTTP API, який дозволяє дистанційно керувати основними портами обладнання (GPIO) і аналого-цифровими перетворювачами (ADC). Для роботи програмованих систем, обладнаних стеком TCP/IP, базові HTTP-запити (протокол передачі гіпертексту) повинні надсилатися до системи. Прикладами таких систем є одноплатні комп'ютери (SBC) і мікроконтролери з підтримкою TCP/IP. Керування пристроєм включає загальний стан і налаштування пристрою, адміністрування портів включає конфігурацію та інформацію про порти, а зворотні виклики API включають виклики API, здійснені за допомогою стандартів зворотних запитів HTTP, які надають широкий спектр функцій і варіантів використання, можуть здаватися дуже складними. Хоча багато з них потрібні для реалізації HAS, більшість є необов'язковими, лише з обмеженим набором функцій, необхідних для його реалізації.

Є багато компонентів екосистеми HAS, включаючи сервер HAS, HASOS (операційна система), а також додаткові пакети та інструменти, адаптовані до різних параметрів. Основним компонентом цієї програми є сервер HAS, створений за допомогою мови Python. Він служить центральним розташуванням, де розміщено зрозумілий онлайн-додаток.

Готова до використання операційна система (ОС для плат Raspberry Pi, HASOS — це сервер, який запускає веб-додаток HAS Server. EspHAS — це модифіковане мікропрограмне забезпечення для пристроїв ESP8266/ESP8285, які використовують HAS API.

Зрештою, додатки — це важливі частини програмного забезпечення, які розширюють можливості пакета програмного забезпечення HAS Server. Пристрої

HAS ідентифікуватимуть себе, включно з їх додатковим набором. Функції та порти, доступні іншим пристроям. Кожен порт визначатиме себе, включаючи його ідентичність, конфігурацію, і т. д. Перед тим, як перейти до наступного кроку топологія мережі досягається інтеграцією головних і підлеглих пристроїв з відносною легкістю.

Wi-Fi або Ethernet – це режими зв’язку. В результаті концентратор дозволить консолідувати адміністрування пристроїв, які використовує HAS (рисунок 2.1).

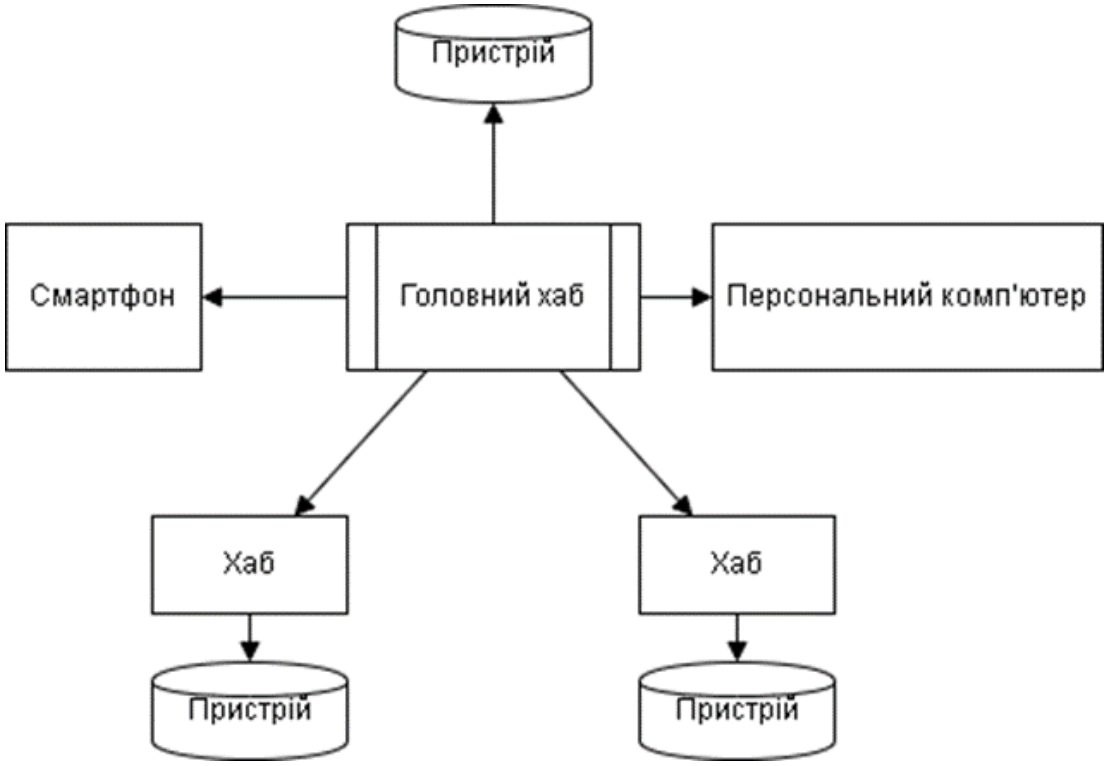


Рисунок 2.1 – Топологія HAS

Концентратори керують користувачем під час спілкування з підключеними пристроями, але інші пристрої також використовують інтерфейс API, який увімкне пристрої. Можна створювати пристрої та концентратори, організовані у складну ієрархію у взаємодії головний-підлеглий. Пристрій може працювати головним для інших підключених пристроїв за допомогою команди HAS. Головний пристрій керує підлеглими пристроями та надає їм доступ через виклики API.

Підлеглий пристрій може так само працювати як головний або головний для додаткових пристроїв. У результаті можна побудувати складні ланцюгові

налаштування головного-підлеглого. Головний може перераховувати, додавати та видаляти підлеглі за допомогою викликів API, ексклюзивних для підлеглих. Головний (головний) пристрій розпізнавав свої підлеглі (суб’єднані) пристрої за допомогою їхніх імен, тоді як головному потрібна була додаткова інформація, щоб імена найслабших пристроїв могли бути змінені в будь-який час. Детальний дизайн показано на рисунку 2.2.

Одна з трьох ролей визначає рівень доступу до пристрою, а найважливіша роль адміністратора, має повний контроль над пристроєм і може змінювати конфігурацію та вигляд; звичайна роль, роль читання-запису, яка може читати з портів і записувати в них, може просто читати значення портів.

Рисунок 2.2 –Блок-схема зв'язку запропонованої HAS

Щоб полегшити автоматизацію, HAS дозволяє створювати правила, які визначають значення портів залежно від різних ситуацій. Вирази можна налаштувати на рівні пристрою або концентратора.

Вирази пристрою швидкі, але вони настільки ж швидкі, наскільки швидкі порти, доступні на пристрої. Це дозволить належним чином реалізувати зв’язки між різними пристроями. Платформа HAS надає три механізми сповіщення для користувачів, які потребують сповіщення про події пристрою, такі як зміна значення порту. У більшості випадків конфігурації HAS використовувалися в приватних мережах, і пристрої не мали доступу до Інтернету. У випадках, коли загальнодоступні IP-адреси доступні, для вирішення проблеми часто використовується переадресація портів. Якщо розширення портів є небажаним або можливим, пристрої можна налаштувати для встановлення з’єднання з загальнодоступним зовнішнім сервером і моніторингу запитів API. Оскільки він дає змогу надсилати HTTP-запити на пристрій, розташований у захищеній мережі, без передачі трафіку порту, цей метод називається зворотним HTTP. З точки зору розробників, HAS надає додаткові модулі, які є простим і швидким засобом об’єднання альтернативних функцій, які часто пов’язані з певним пристроєм. Додатки можуть бути загальнодоступними або приватними відповідно до вимог розробника та обмежень ліцензіата на програмне забезпечення.

Безпека HAS забезпечується впровадженням найкращих методів програмування в сучасних Інтернет-додатках. Нелегальний користувач хоче спілкуватися з Хабом поза межами; HTTPS захищає зв'язок. Це гарантує, що зв’язок HTTP зашифрований, концентратор є законним і повідомлення не пошкоджені. Звичайний HTTP використовується лише локально, наприклад, у будівлях тощо, для зв’язку між концентраторами та пристроями, що знаходяться під їх контролем. Сертифікат TLS допомагає досягти перелічених вище цілей безпеки в поєднанні з HTTPS. Шифрування відповідає за створення та оновлення сертифікатів TLS. Коли термін дії сертифіката наближається, ця операція автоматично запускається на концентраторі. HAS використовується для віддаленого доступу до Hub (з адміністративною метою).

Протокол HAS використовує ECDSA (або відповідний алгоритм) для автентичного шифрування інформації для входу. Також можна використовувати онлайн-логін, пароль адміністратора та ім’я користувача, встановлені на хабі. Три

правила API визначають права, надані для запитів API: звичайний користувач, користувач лише для перегляду та адміністратор. Кожна роль має власний набір дозволів. Запити API надають дані автентифікації через веб-токен JSON (JWT), визначений RFC 7519. Використання спільного секрету (також відомого як пароль) забезпечує легітимність абонента. Замість цього ми могли використати дайджест HTTP, базову автентифікацію HTTP або метод керування сеансом, який покладається на файли cookie та поєднує їх із звичайною формою входу. У той час як базова автентифікація є вразливою, коли надходить через незашифровані лінії, дайджест-автентифікація є надто складною та потребує обміну численними повідомленнями, що є непотрібним. Техніка на основі файлів cookie/сеансу сприйнятлива до атак крадіжки сеансів і може бути небезпечною, якщо надсилати її через незашифровані мережі. У разі виявлення вразливості вбудований механізм бездротового зв'язку (OTA) (оновлення мікропрограми) гарантує, що концентратор і пов'язані з ним пристрої завжди працюють із найновішою доступною версією, що дозволяє нам негайно розгорнути безпеку. Це показує, що користувач і адміністратор, які використовують запропонований HAS, мають програму, встановлену на своєму смартфоні та підключену до інтернет-служб за допомогою головного концентратора. Головний концентратор включає комунікаційні пристрої, такі як модуль Raspberry Pi з локальною мережею (LAN) і локальною базою даних. Локальна база даних зберігає записи користувачів, інформацію для входу та дані панелі інструментів. Крім того, головний концентратор зв'язується з пристроєм Wi-Fi (ESP8266) для контролю та моніторингу побутової техніки, такої як вентилятор, кондиціонер, телевізор, проектор тощо, у всіх кімнатах, включаючи вітальні, спальні та їдальні. HAS також розрахував одиниці споживання за день, місяць і рік на основі побутової техніки.

2.2 Алгоритмічне забезпечення системи

Базуючись на архітектурі системи яка була запропонована раніше розумна система автоматизації будинку працює за досить простою логікою, яка складається

з кількох етапів. Робота стартує із збору даних. Спеціальні датчики, що встановлені у будинку, постійно фіксують важливі параметри такі як температура повітря, рівень освітлення, рух у кімнаті та вологість. Ці дані передаються на центральний пристрій – Raspberry Pi, який виступає як головний контролер системи.

На Raspberry Pi вся зібрана інформація обробляється. Система аналізує ці дані (рисунок 2.3) та порівнює їх з налаштованими порогами. Наприклад, якщо температура перевищує 26°C, система ввімкне кондиціонер.

Після цього на основі аналізу приймається рішення. Якщо умови виконуються, Raspberry Pi відправляє команду на відповідний пристрій (модуль ESP8266), який підключений до кондиціонера або лампи.

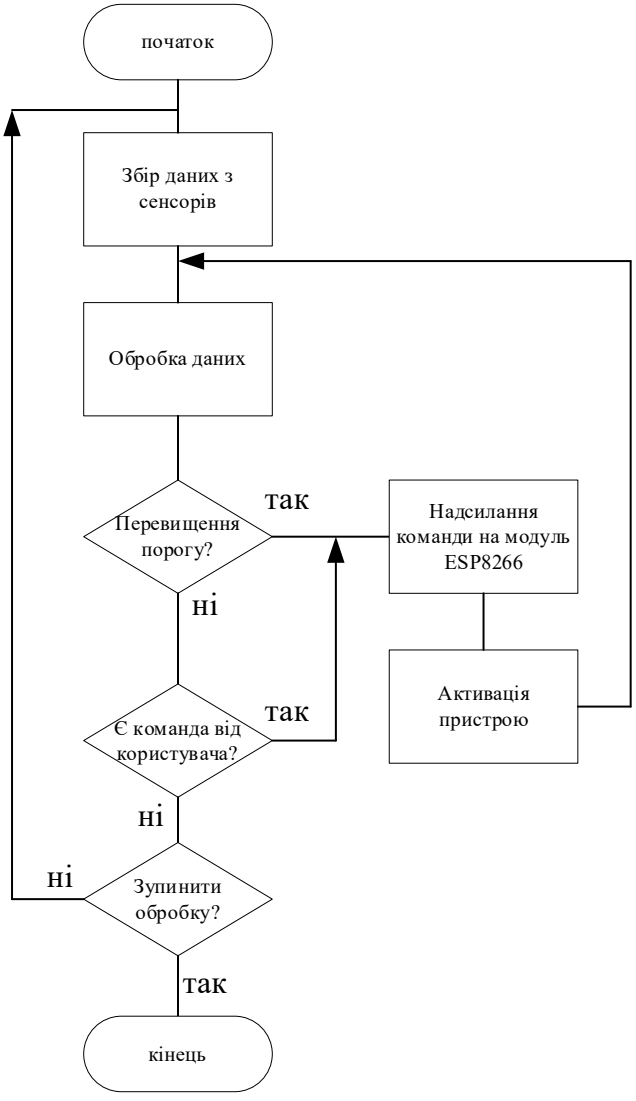


Рисунок 2.3 –Схема алгоритму функціонування системи

На останньому етапі відбувається виконання дії — ESP8266 вмикає або вимикає пристрій. Уся ця логіка може працювати автоматично за правилами, так і з участю користувача.

Користувач має змогу керувати пристроями або переглядати їхній стан через мобільний додаток Android, який підключається до Raspberry Pi. Якщо користувач хоче змінити параметри або увімкнути пристрій вручну — додаток відправляє команду через Wi-Fi, і вона також виконується через ESP.

Так як основна задача розумного будинку реагувати на зміни у навколишньому середовищі. Для цього система використовує сенсори, які зчитують різні параметри: температуру повітря, рівень вологості, наявність руху в кімнаті, рівень освітлення тощо (рисунок 2.4).

Рисунок 2.4 – Алгоритм взаємодії з сенсорами та виконавчими пристроями

Ці сенсори підключаються до мікроконтролера, наприклад ESP32-P4R через спеціальні входи — GPIO-порти.

Деякі сенсори, такі як температурні або вологості, передають дані у вигляді аналогового сигналу. Цей сигнал потрібно перетворити на цифрове значення, яке система може обробити. Це робиться за допомогою вбудованого аналогово-цифрового перетворювача (ADC).

Після отримання цифрового значення система порівнює його з пороговими значеннями, які визначають, чи потрібно виконати якусь дію. Виконавчі пристрої - реле, лампи, вентилятори, помпи отримують команди через той самий мікроконтролер. Команда може бути подана автоматично або після підтвердження користувачем через мобільний застосунок.

Основу системи становить Raspberry Pi, який виконує роль головного хаба — тобто центру, що приймає дані від усіх пристроїв. До нього по Wi-Fi підключаються модулі ESP32-P4, які встановлені біля різних сенсорів і передають інформацію про температуру, рух, вологість, споживання електроенергії тощо (рисунок 2.5).

Рисунок 2.5 – Алгоритм обробки та зберігання даних на головному концентраторі

Кожен модуль ESP32-P4 періодично надсилає HTTP-запити до Raspberry Pi. Ці запити передаються у вигляді структурованих даних (наприклад, JSON). Raspberry Pi, у свою чергу, має програму-сервер, яка отримує ці запити, обробляє їх і витягує потрібну інформацію.

Після отримання даних Raspberry Pi зберігає їх у локальній базі даних, наприклад, SQLite. Кожен запис зберігає значення сенсора, час отримання, назву кімнати або пристрою. Це дозволяє згодом будувати графіки, аналізувати історію показників або розраховувати витрати енергії.

Також система формує лог-файли — тобто журнали подій і дій, які можуть бути використані для перегляду, що відбувалося в системі за певний період. Наприклад, можна побачити, коли вмикався кондиціонер, скільки електроенергії було спожито за добу, місяць або рік.

Для зручності користувача, Raspberry Pi підготовлює ці дані до відображення на веб-панелі або в мобільному застосунку. Усі показники виводяться у вигляді таблиць, графіків або повідомлень.

В системі домашньої автоматизації мобільний застосунок на Android є головним інструментом взаємодії користувача з пристроями. Саме через додаток користувач має змогу переглядати стан системи, вмикати або вимикати пристрої, змінювати налаштування, а також отримувати повідомлення про події.

Коли користувач натискає кнопку у додатку — наприклад, щоб увімкнути світло в кімнаті — мобільний застосунок надсилає керуючу команду через HTTP-запит або REST API до головного пристрою системи — Raspberry Pi. Цей пристрій отримує команду та пересилає її далі до мікроконтролера ESP32-P4, який фізично керує виконавчим елементом (наприклад, реле для вмикання світла).

Після виконання дії система оновлює стан пристрою і надсилає відповідь назад у додаток. Додаток одразу відображає новий статус — наприклад, змінює піктограму лампи з "вимкнено" на "увімкнено".

Також система може ініціювати зворотній зв’язок, коли стається певна подія: рух, перевищення температури, низький рівень води тощо. У таких випадках

Raspberry Pi відправляє push-повідомлення на смартфон користувача або оновлює інтерфейс у реальному часі (рисунок 2.6).

Рисунок 2.6 – Алгоритм взаємодії мобільного додатку з системою

Мобільний застосунок забезпечує зручний, швидкий та зрозумілий спосіб керування всією системою та отримання сповіщень.

Ще однією з важливих функцій розумної системи також є автоматичне виконання дій без участі користувача. Для цього в системі задаються сценарії правила, які система перевіряє постійно. Як тільки певна умова виконується система миттєво реагує.

Такі сценарії будуються на основі умовної логіки, відомої як IF–THEN–ELSE (рисунок 2.7):

– Якщо (IF) система виявляє рух у кімнаті, то (THEN) увімкнути світло;

– Інакше (ELSE) — не робити нічого або вимкнути освітлення після певного часу.

Рисунок 2.7 – Алгоритми автоматичних сценаріїв і правил

Для цього система постійно зчитує значення з датчиків, підключених до портів GPIO. Коли значення порту змінюється (наприклад, сигнал про рух), система запускає тригер (trigger), який перевіряє умову і виконує відповідну дію.

Це дозволяє системі працювати автономно, без потреби у втручанні людини, забезпечуючи комфорт і економію ресурсів. Користувач може змінювати або створювати нові сценарії через мобільний застосунок або веб-інтерфейс.

2.3 Інформаційне забезпечення

В розумній системі домашньої автоматизації, яка працює на базі Інтернету речей (IoT), вся робота базується на передачі та обробці інформації. Умовно всю інформацію можна поділити на вхідну та вихідну.

Вхідна інформація — це ті дані, які система отримує ззовні. Це можуть бути показники з різних датчиків: температура повітря в кімнаті, вологість, рівень освітлення, наявність руху тощо. Дані надходять у цифровому вигляді і постійно оновлюються, щоби система знала поточний стан середовища. Також до вхідної інформації належать команди користувача, які він надсилає зі свого смартфона — увімкнути світло, змінити температурний режим чи налаштувати розклад. Крім цього, система може отримувати внутрішні сигнали, такі як таймери, сигнали від кнопок, сповіщення про збої або помилки (рисунок 2.8).

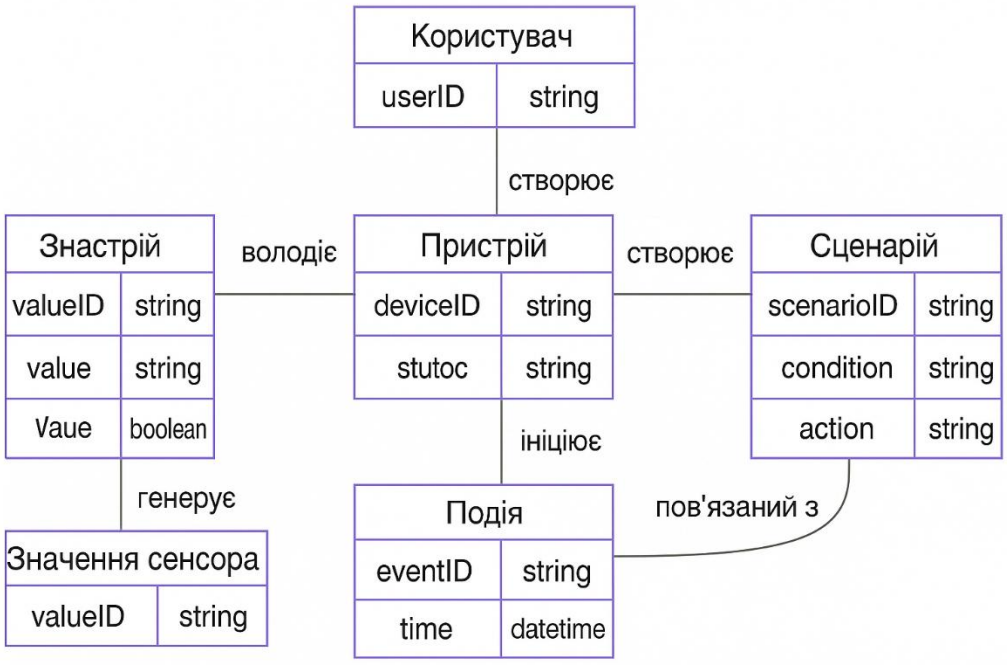


Рисунок 2.8– Інфологічна модель системи

Вихідна інформація — це дані, які система повертає або показує користувачу. Це може є стан приладів чи працює кондиціонер, чи ввімкнене освітлення, чи зачинені двері. Система також формує журнали подій, де записуються всі дії, що

сталося, коли і за яких умов. Ще один важливий тип вихідної інформації повідомлення на телефон користувача, сповіщення про рух у кімнаті або повідомлення про надто високу температуру. Також система може формувати статистику скільки електроенергії було витрачено за день або тиждень.

Нижче описано інфологічна модель системи та її сутності.

– User взаємодіє з Device (1:N) — користувач володіє одним або кількома пристроями.

– Device генерує SensorValue (1:N) — сенсорні пристрої передають багато значень.

– SensorValue може ініціювати Event (N:1) — подія на основі перевищення порогу.

– User створює Scenario (1:N) — кожен користувач може мати набір сценаріїв.

– Scenario пов'язаний з Device (N:M) — сценарії застосовуються до пристроїв

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Вибір та аналіз програмно-апаратних засобів для розробки системи

Для побудови розумної системи домашньої автоматизації було обрано мікроконтролер ESP32-P4 (рисунок 3.1), як основний пристрій для взаємодії з сенсорами та виконавчими механізмами.

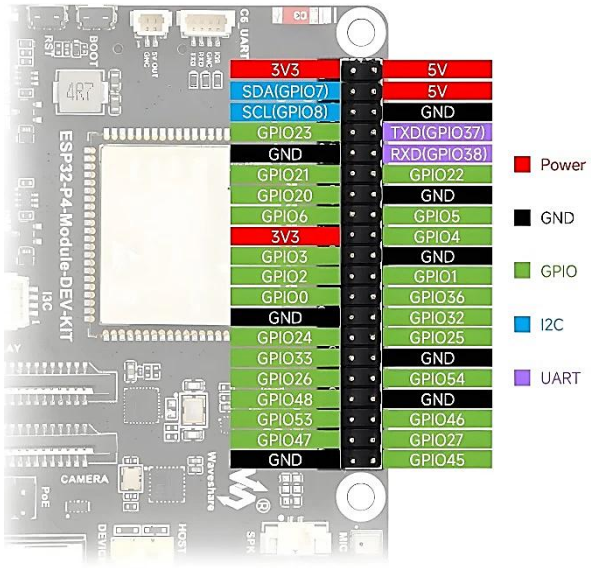


Рисунок 3.1 – Модуль розробки ESP32-P4

Цей модуль забезпечує стабільну роботу, має вбудований Wi-Fi, підтримує цифрові та аналогові входи/виходи, а також дозволяє реалізовувати мережеві запити через HTTP. Такий вибір зумовлений його потужністю, низьким енергоспоживанням.

В якості центрального елемента системи, до якого надходять дані з усіх периферійних пристроїв, використано Raspberry Pi 4B. Цей мікрокомп’ютер виконує функції локального сервера, обробляє запити, зберігає дані в базі даних, а також надає інтерфейс для доступу користувача через мобільний додаток або вебінтерфейс. Він має достатню обчислювальну потужність, підтримує Wi-Fi, Ethernet, GPIO, і є придатним для запуску серверних додатків на мові Python.

До системи підключаються наступні сенсори:

– датчик температури (наприклад, DHT22),

– датчик вологості,

– інфрачервоний датчик руху (PIR),

– аналоговий датчик освітленості.

Ці пристрої дозволяють отримувати дані про стан навколишнього середовища та керувати пристроями на основі отриманої інформації.

На етапі тестування та розробки мікроконтролерної логіки було використано Wokwi (рисунок 3.2) - онлайн-симулятор для Arduino та ESP32. Він дозволяє моделювати роботу сенсорів, мікроконтролерів, реле й інших елементів без фізичного обладнання.

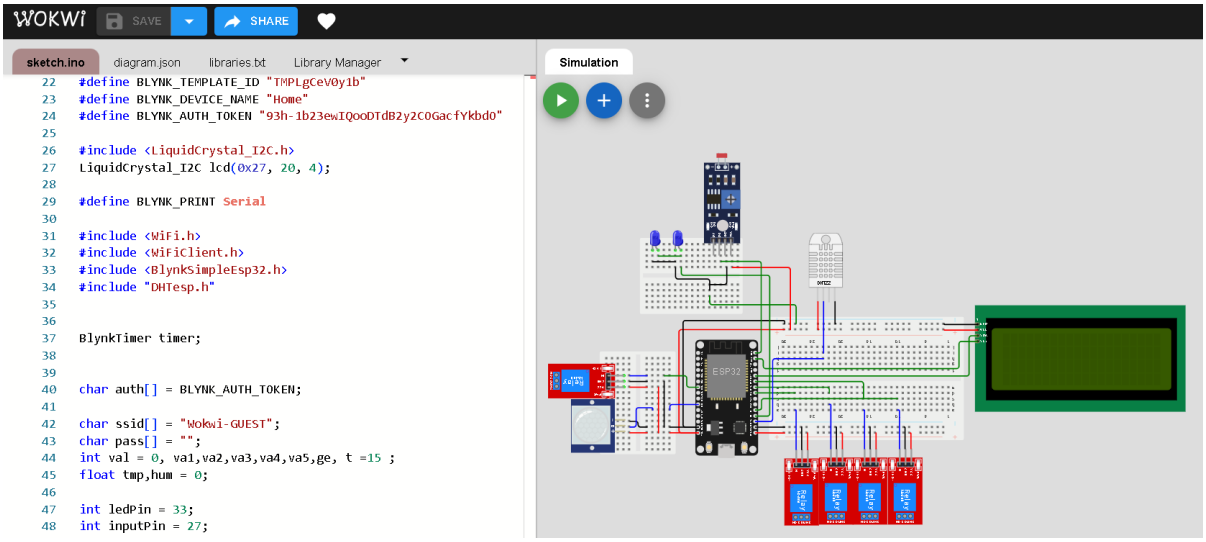


Рисунок 3.2 – Середовище розробки wokwi.com

Це дає змогу відлагодити програмну логіку, перевірити роботу сценаріїв та провести попереднє тестування без витрат на апаратні ресурси. У середовищі було змодельовано конфігурації типу сенсор → ESP32 → світлодіод та реле.

Програмна частина реалізована на мові Python з використанням мікрофреймворку Flask, що дозволяє обробляти HTTP-запити та організовувати REST API. Для зберігання даних обрана база даних SQLite, що забезпечує простоту використання, не потребує розгортання окремого серверу та підходить для невеликих IoT-систем. Така структура програмного забезпечення дозволяє легко масштабувати систему, додавати нові сенсори або пристрої, а також інтегрувати інші інтерфейси.

3.2 Програмно технологічне забезпечення системи

В даному розділі описано основні програмні модулі та їх функціонал. Нижче представлено код для мікроконтролера ESP32 -P4, який зчитує температуру з датчика DHT22 та надсилає її на сервер за допомогою інтернет-з'єднання.

На початку задаються параметри датчика: пін, до якого він підключений (DHTPIN 4), та його тип (DHT22). Створюється об'єкт dht для взаємодії з датчиком (рисунок 3.3).

```
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

Рисунок 3.3 – Налаштування пінів датчиків

Далі задаються дані для підключення до Wi-Fi (рисунок 3.4).

```
const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";
const char* serverName = "http://<RASPBERRY_PI_IP>:5000/api/sensor";
```

Рисунок 3.4 – Налаштування Wi-Fi

Далі ініціалізується послідовний порт та сам датчик, після чого ESP32 підключається до Wi-Fi (рисунок 3.5).

```
void setup() {
  Serial.begin(115200);
  dht.begin();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("З'єднання з Wi-Fi...");
  }
  Serial.println("Підключено!");
}
```

Рисунок 3.5 – Налаштування Wi-Fi

Кожні 10 секунд ESP32 зчитує температуру з датчика, якщо значення коректне (не NaN), формується HTTP POST-запит String payload = "{\"device_id\": \"esp32-1\", \"temperature\": " + String(temp) + "}". Далі використовуючи бібліотеку HTTPClient, ESP32 надсилає дані у форматі JSON на сервер Flask, розгорнутий на сервері (рисунок 3.6).

```
HTTPClient http;
http.begin(serverName);
http.addHeader("Content-Type", "application/json");
int httpResponseCode = http.POST(payload);
```

Рисунок 3.6 – Відправка JSON запиту на сервер

У відповідь ESP32 виводить код відповіді у послідовний порт (рисунок 3.7)

```
Serial.print("Код відповіді: ");
Serial.println(httpResponseCode);
```

Рисунок 3.7 – Комунікація через послідовний порт

Після цього з’єднання закривається, і через 10 секунд цикл повторюється. Таким чином, ESP32 періодично вимірює температуру і передає її на сервер для подальшої обробки або відображення у веб-інтерфейсі.

Для комунікації з апаратною частиною було реалізовано бекенд код. Він реалізує простий вебсервер на Flask, який працює на Raspberry Pi та приймає дані про температуру та команди від мікроконтролера ESP32 та мобільного додатку. Дані зберігаються в локальній базі даних SQLite (рисунок 3.8).

```
from flask import Flask, request, jsonify
import sqlite3
from datetime import datetime
app = Flask(__name__)
def save_to_db(device_id, temperature):
    conn = sqlite3.connect("iot.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS sensors (id INTEGER PRIMARY KEY AUTOINCREMENT, device_id TEXT, temperature REAL, timestamp TEXT)")
    cursor.execute("INSERT INTO sensors (device_id, temperature, timestamp) VALUES (?, ?, ?)",
                   (device_id, temperature, datetime.now().isoformat()))
    conn.commit()
    conn.close()
@app.route('/api/sensor', methods=['POST'])
def sensor_data():
    data = request.get_json()
    save_to_db(data["device_id"], data["temperature"])
    return jsonify({"status": "OK"})
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Рисунок 3.8 – Вебсервер на Flask

На рисунку 3.9 показано розмітка інтерфейсу (activity_main.xml).

```
<WebView
    android:id="@+id/webView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Рисунок 3.9 – Розмітка інтерфейсу

Даний елемент WebView є вікно всередині додатку, яке показує веб-сторінку так само, як у браузері. Весь екран буде зайнятий веб-інтерфейсом. Також на рисунку 3.10 представлено основний код активності (MainActivity.kt).

```
val webView: WebView = findViewById(R.id.webView)
webView.settings.javaScriptEnabled = true
webView.webViewClient = WebViewClient()
webView.loadUrl("http://192.168.1.100:5000")
```

Рисунок 3.10 – MainActivity.kt

– findViewById знаходить компонент WebView за ID.

– JavaScriptEnabled = true дозволяє виконувати JavaScript у веб-сторінці.

– WebViewClient() не відкривати посилання в браузері, а залишатись в додатку.

– loadUrl(...) завантажує веб-сторінку з Flask-сервера, яка містить кнопки керування реле.

Далі починає працювати веб сторінка (рисунок 3.11).

```
<button onclick="sendCommand('relay1','on')">Увімкнути реле</button>
<button onclick="sendCommand('relay1','off')">Вимкнути реле</button>
```

Рисунок 3.11 – Елементи кнопок включення реле

Коли користувач натискає кнопку, викликається JavaScript-функція (рисунок 3.12).

```
function sendCommand(device, state) {
  fetch("http://192.168.1.100:5000/api/relay", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ device_id: device, state: state })
  })
}
```

Рисунок 3.12 – JavaScript-функція виклику

Вона надсилає HTTP POST-запит на Flask-сервер із JSON-даними {"device_id": "relay1", "state": "on"}.

3.3 Тестування системи

Після того, як всі компоненти системи були описані було проведено тестування системи. Інтерфейс системи відкривається в додатку Android через компонент WebView, який відображає просту вебсторінку з двома кнопками, «Увімкнути реле» та «Вимкнути реле» (рисунок 3.13). Кожна з цих кнопок відповідає за надсилання певної команди на сервер.

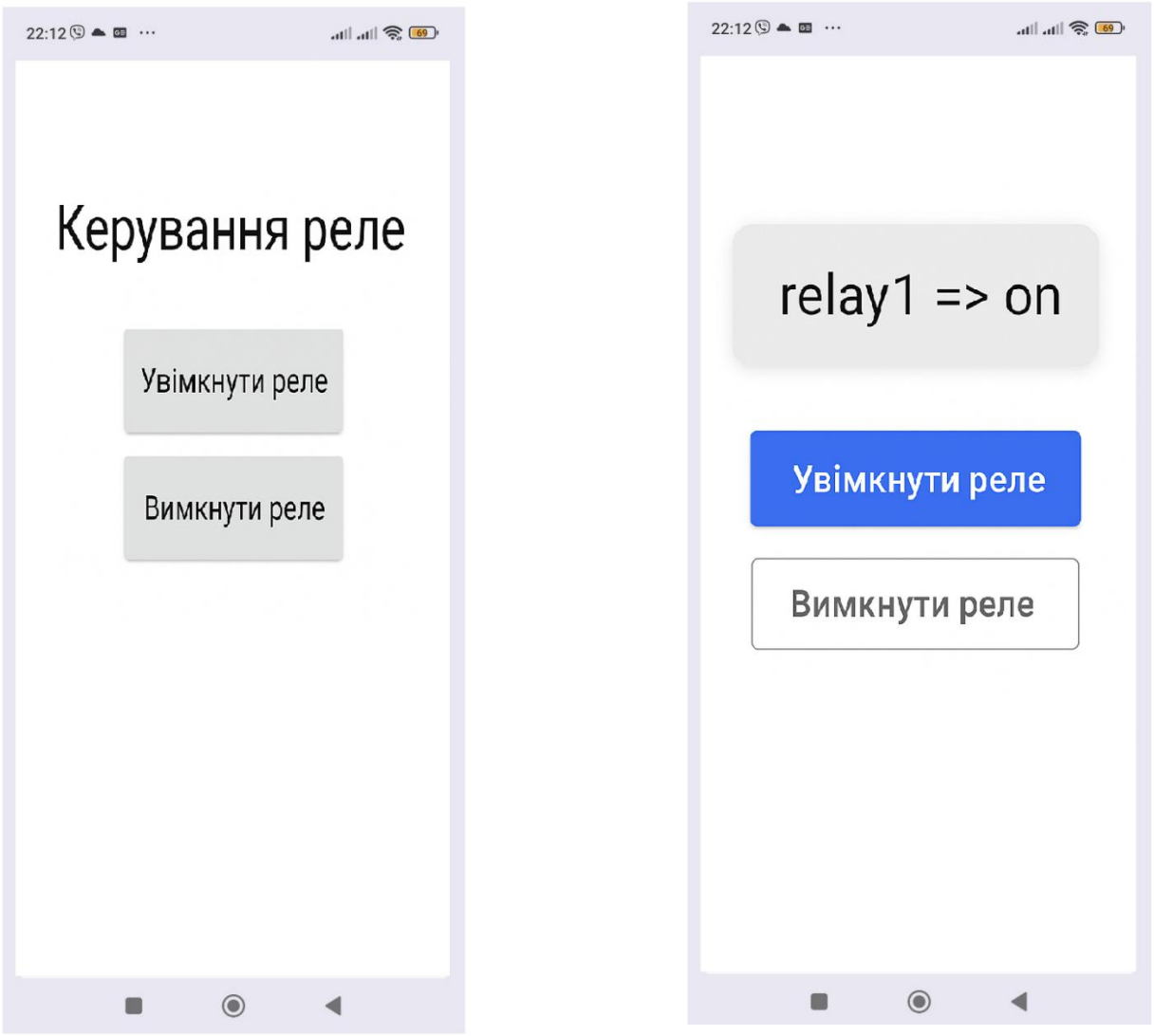


Рисунок 3.13 – Тестування включення та виключення реле

Також додаток відображає графік температури з 40 попередніми точками. Крім того відображається статус кондиціонера залежно від температури (вмикається, якщо температура > 25.5 °C).

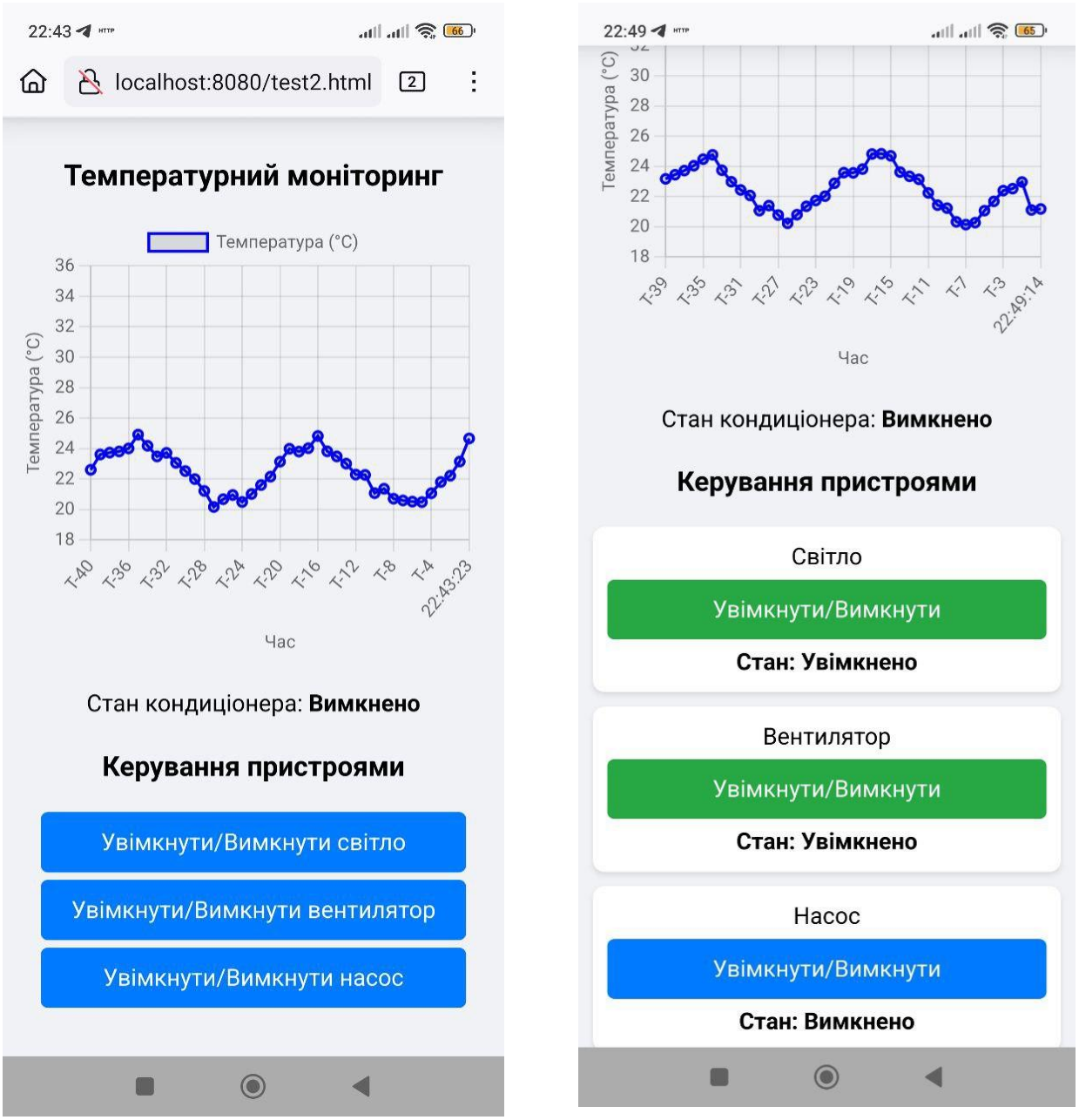


Рисунок 3.14 – Тестування Управління пристроями

Тестування системи керування реле через мобільний вебінтерфейс здійснювалося в кілька етапів. Спочатку було перевірено відображення інтерфейсу на мобільному пристрої за допомогою WebView та мобільного браузера.

Далі проведено функціональні тести взаємодії з кнопками керування. Було перевірено, що після натискання кнопок «Увімкнути/Вимкнути» змінюється їхній колір, оновлюється текстовий статус і відображається правильне повідомлення про стан пристрою. Також моделювалися ситуації втрати підключення, перевірялась реакція інтерфейсу на оновлення даних з графіка, і тестувалась стабільність системи при багаторазовому перемиканні. Усі перевірки підтвердили, що система є зручною, стабільною й інтуїтивно зрозумілою для користувача.

ВИСНОВКИ

При виконанні кваліфікаційної роботи розроблено та протестовано інтелектуальну систему автоматизації житла на основі технологій Інтернету речей (IoT) з використанням мобільного застосунку на платформі Android. Система забезпечує моніторинг стану навколишнього середовища та інтелектуальне керування побутовими пристроями в режимі реального часу.

1. Проведено огляд технологій Інтернету речей та домашньої автоматизації.

2. Проведено огляд і аналіз відкритих та комерційних платформ розумного дому.

3. Сформульовано постановку задачі та поставлено мету розробити енергоефективну, локально захищену та масштабовану систему, що не залежить від зовнішніх хмарних сервісів та керується через Android-додаток.

4. Розроблено алгоритмічне та інформаційне забезпечення системи домашньої автоматизації на базі Android та IoT.

5. Запропоновано трирівневу архітектуру системи.

6. Розроблено алгоритми оброблення подій (IF-THEN-ELSE правила), сценарне керування та механізм OTA-оновлень мікропрограм .

7. Побудовано інфологічну модель та журнальну підсистему де дані зберігаються в SQLite, що задовольняє вимоги до локального збереження інформації.

8. Обґрунтовано вибір Raspberry Pi 4 B за критеріями продуктивності й універсальності GPIO та ESP32-P4 як енергоощадного вузла збору даних. Досліджено набір сенсорів і показано, що обрана конфігурація покриває базові сценарії клімат-, світло- й безпекового контролю.

9. Реалізовано програмно-технологічну реалізацію на базі Flask-сервера з REST-API; мобільний Android-додаток з використанням WebView-обгортки для керування та візуалізації графіків температури.

10. Виконано функціональні тести перемикання реле, реакції на втрату зв’язку та стрес-тести інтерфейсу. Тести показали, що система стабільно обробляє події, графічно відображає стан і дає коректний зворотний зв’язок.

Розроблена система є доступним, ефективним та масштабованим рішенням для реалізації концепції «розумного дому».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Debnath B., Dey R., Roy S. Smart Switching System Using Bluetooth Technology. 2019 *Amity International Conference on Artificial Intelligence* (AICAI), Dubai, United Arab Emirates, 2019. P. 760–763. DOI: 10.1109/AICAI.2019.8701298

2. Al Razib M., Javeed D., Khan M.T., Alkanhel R., Muthanna M.S.A. Cyber Threats Detection in Smart Environments Using SDN-Enabled DNN-LSTM Hybrid Framework. *IEEE Access*. 2022. Vol. 10. P. 53015–53026.

3. Kumar A. A. Novel Decentralized Blockchain Architecture for the Preservation of Privacy and Data Security against Cyberattacks in Healthcare. *Sensors.* 2022. Vol. 22. P. 5921.

4. Murthy A. Internet of Things, a vision of digital twins and case studies IoT and Spacecraft Informatics. Amsterdam: *Elsevier*, 2022. P. 101–127.

5. Baraka K. Low Cost Arduino Android-Based Energy-Efficient Home Automation System with Smart Task Scheduling. *Proceedings of the 2013 5th International Conference on Computational Intelligence, Communication Systems and Networks*, Madrid, Spain. 5–7 June 2013. P. 296–301.

6. Stolojescu-Crisan, Cristina, Calin Crisan, and Bogdan-Petru Butunoi. An IoT-based smart home automation system. *Sensors* 21.11. 2021 3784.

7. Singh M., Dhablia, A. Smart home automation using iot: Prototyping and integration of home devices. *Research Journal of Computer Systems and Engineering*, 4(2)., P. 130-143.

8. Liang, Chua Boon. Smart home security system based on Zigbee. In: Advances in Smart System Technologies: Select Proceedings of ICFSST 2019. *Springer Singapore*, 2021. P. 827-836.

9. Ilyas, M., Uçan, O., El Mohamad, Y. Smart home automation system design based on IoT device cloud. *ICMI 2021*, P.116-123.

10. Javeed D. A hybrid deep learning-driven SDN enabled mechanism for secure communication in Internet of Things (IoT). *Sensors*. 2021. – Vol. 21. – P. 4884.

11. Majeed R. An Intelligent, Secure, and Smart Home Automation System. *Scientific Programming*. 2020. Vol. 2020. Article ID 4579291.

12. Khan M.T., Akhunzada A., Zeadally S. Proactive Defense for Fog-to-Things Critical Infrastructure. IEEE *Communications Magazine*. 2022.

13. Stolojescu-Crisan C. IoT based intelligent building applications in the context of COVID-19 pandemic. *Proceedings of the 2020 International Symposium on Electronics and Telecommunications (ISETC)*, Timisoara, Romania. 5–6 November 2020.

14. Krichen M. A formal testing model for operating room control system using internet of things. *Computer Materials & Continua*. 2021. Vol. 66. P. 2997–3011.

15. openHAB. openHAB. URL: http://www.openhab.org/ (дата звернення: 12.03.2025)

16. Home Assistant. Home Assistant. URL: http://www.home-assistant.io/ (дата звернення: 12.03.2025).

17. Ali S. A blockchain-based secure data storage and trading model for wireless sensor networksю International Conference on Advanced Information Networking and Applications, Caserta, Italy, 15–17 April 2020. *Springer*: Cham, Switzerland, 2020.

18. Calaos, Open Source Home Automation. URL: https://calaos.fr/en/ (дата звернення: 12.03.2025).

19. Fernández-Caramés T.M. An Intelligent Power Outlet System for the Smart Home of the Internet of Things. *International Journal of Distributed Sensor Networks*. 2015. – Vol. 11. Article ID 214805.

20. Jeedom Official Web Page URL: https://www.jeedom.com/ (дата звернення: 12.06.2025).

21. Shahajan M. Internet of Things (IoT) based automatic electrical energy meter billing system. *Journal of Electronics and Communication Engineering*. 2019., Vol. 14., P. 39–50.

22. Islam U. Detection of Distributed Denial of Service (DDoS) Attacks in IoT Based Monitoring System of Banking Sector Using Machine Learning Models. *Sustainability*. 2022. Vol. 14. – P. 8374.

23. Singh S., Sharma P.K., Park J.H. SH-SecNet: An enhanced secure network architecture for the diagnosis of security threats in a smart home. *Sustainability*. – 2017. Vol. 9., P. 513.

24. Choi W. Smart home and internet of things: A bibliometric study. *Journal of Cleaner Production*. 2021. Vol. 301. Article ID 126908.

25. Triantafyllou A., Sarigiannidis P., Lagkas T.D. Network protocols, schemes, and mechanisms for Internet of Things (IoT): Features, open challenges, and trends. *Wireless Communications and Mobile Computing*. 2018. Article ID 5349894.

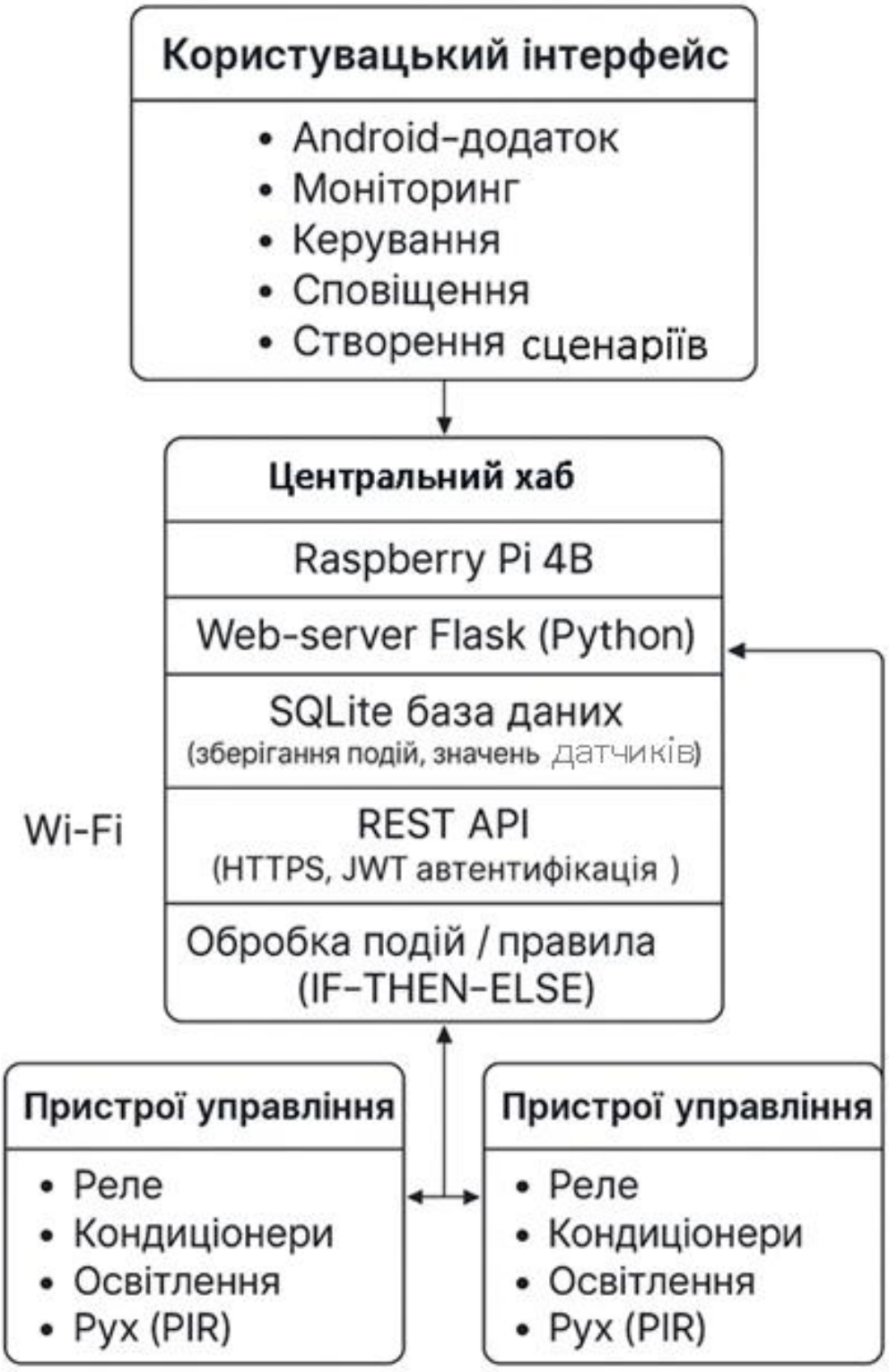
26. Мельник Н., Осолінський О. Архітектура системи домашньої автоматизації на базі android з використанням інтернету речей Інтелектуальні інформаційні технології в прикладних дослідженнях : тези доп. студентської наук.-практ. конф. (м. Тернопіль, 27–29 травня 2025 р.). Тернопіль, 2025. С.306-308

27. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

28. Островерхов В. М., Біловус Л. І., Возьний К. З., Луцишин О. О., Монастирський Г. Л., Надвиничний С. А., Питель С. В., Шандрук С. К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

Додаток А

Архітектура розумної системи домашньої автоматизації на базі Android



Додаток Б

Код для Flask-сервера

```
from flask import Flask, request, jsonify
import sqlite3
from datetime import datetime

app = Flask(__name__)

def save_to_db(device_id, temperature):
    conn = sqlite3.connect("iot.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS sensors (id INTEGER PRIMARY KEY AUTOINCREMENT, device_id TEXT, temperature REAL, timestamp TEXT)")
    cursor.execute("INSERT INTO sensors (device_id, temperature, timestamp) VALUES (?, ?, ?)",
                   (device_id, temperature, datetime.now().isoformat()))
    conn.commit()
    conn.close()

@app.route('/api/sensor', methods=['POST'])
def sensor_data():
    data = request.get_json()
    save_to_db(data["device_id"], data["temperature"])
    return jsonify({"status": "OK"})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Додаток В

Код програми для мобільного додатку

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Моніторинг температури та керування</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
    body {
      font-family: sans-serif;
      margin: 0;
      padding: 10px;
      background: #f0f2f5;
    }
    h2 {
      text-align: center;
      font-size: 20px;
    }
    .status, .controls {
      margin-top: 20px;
      text-align: center;
      font-size: 16px;
    }
    .device-control {
      margin: 10px auto;
      padding: 10px;
      max-width: 320px;
      background-color: #fff;
      border-radius: 8px;
      box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    }
    .device-control button {
      padding: 10px;
```

```
      margin-top: 5px;
      font-size: 16px;
      width: 100%;
      border: none;
      border-radius: 5px;
      background-color: #007bff;
      color: white;
      cursor: pointer;
      transition: background-color 0.3s ease;
    }
    .device-control button.active {
      background-color: #28a745;
    }
    .device-status {
      margin-top: 5px;
      font-weight: bold;
    }
    .chart-container {
      position: relative;
      width: 100%;
      max-width: 600px;
      height: 300px;
      margin: 0 auto;
    }
    canvas {
      width: 100% !important;
      height: 100% !important;
    }
  </style>
</head>
<body>
  <h2>Температурний моніторинг</h2>
  <div class="chart-container">
    <canvas id="tempChart"></canvas>
  </div>
```

```
  <div class="status">
    Стан кондиціонера: <strong id="acStatus">Очікується...</strong>
  </div>

  <div class="controls">
    <h3>Керування пристроями</h3>

    <div class="device-control">
      <div>Світло</div>
      <button id="btn-light"
onclick="toggleDevice('light')">Увімкнути/Вимкнути</button>
      <div class="device-status" id="status-light">Стан:
Вимкнено</div>
    </div>

    <div class="device-control">
      <div>Вентилятор</div>
      <button id="btn-fan"
onclick="toggleDevice('fan')">Увімкнути/Вимкнути</button>
      <div class="device-status" id="status-fan">Стан: Вимкнено</div>
    </div>

    <div class="device-control">
      <div>Насос</div>
      <button id="btn-pump"
onclick="toggleDevice('pump')">Увімкнути/Вимкнути</button>
      <div class="device-status" id="status-pump">Стан:
Вимкнено</div>
    </div>
  </div>

  <script>
    const ctx =
document.getElementById('tempChart').getContext('2d');
```

```
    const initialData = Array.from({ length: 40 }, (_, i) => 22 +
Math.sin(i / 3) * 2 + Math.random());
    const initialLabels = Array.from({ length: 40 }, (_, i) => `T-${40
- i}`);
    const initialACStatus = true;

    const tempChart = new Chart(ctx, {
      type: 'line',
      data: {
        labels: initialLabels,
        datasets: [{
          label: 'Температура (°C)',
          data: initialData,
          borderColor: 'blue',
          borderWidth: 2,
          fill: false
        }]
      },
      options: {
        responsive: true,
        maintainAspectRatio: false,
        animation: false,
        scales: {
          x: {
            title: { display: true, text: 'Час' }
          },
          y: {
            title: { display: true, text: 'Температура (°C)' },
            suggestedMin: 18,
            suggestedMax: 35
          }
        }
      }
    });
```

```
    document.getElementById('acStatus').textContent = initialACStatus
? 'Увімкнено' : 'Вимкнено';

    setInterval(() => {
      const newTemp = 22 + Math.sin(Date.now() / 10000) * 2 +
Math.random();
      const newTime = new Date().toLocaleTimeString();

      if (tempChart.data.labels.length > 40) {
        tempChart.data.labels.shift();
        tempChart.data.datasets[0].data.shift();
      }

      tempChart.data.labels.push(newTime);
      tempChart.data.datasets[0].data.push(newTemp);
      tempChart.update();

      const acOn = newTemp > 25.5;
      document.getElementById('acStatus').textContent = acOn ?
'Увімкнено' : 'Вимкнено';
    }, 3000);

    const deviceStates = {
      light: false,
      fan: false,
      pump: false
    };

    function toggleDevice(device) {
      deviceStates[device] = !deviceStates[device];
      const button = document.getElementById('btn-' + device);
      const status = document.getElementById('status-' + device);

      if (deviceStates[device]) {
        button.classList.add('active');
        status.textContent = 'Стан: Увімкнено';
```

```
      } else {
        button.classList.remove('active');
        status.textContent = 'Стан: Вимкнено';
      }

      // fetch(`/api/device/${device}`, { method: 'POST' });
    }
  </script>
</body>
</html>
```

Додаток Г
Апробація отриманих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(IITAR-2025)

27-29 травня 2025 року

Тернопіль

2025

Збірник тез доповідей студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR-2025). (Тернопіль, 27-29 травня 2025 року). Тернопіль: ЗУНУ, 2025. 372 с.

До збірника увійшли тези доповідей учасників студентської науково-практичної конференції "Інтелектуальні інформаційні технології в прикладних дослідженнях" (IITAR – 2025), що відбувалась у рамках AI Week на базі кафедри інформаційно-обчислювальних систем і управління Західноукраїнського національного університету. Мета конференції — об'єднати науковців, освітян, представників IT-бізнесу та здобувачів освіти для обміну досвідом, презентації сучасних досліджень і впровадження інноваційних рішень у сфері комп'ютерних наук, штучного інтелекту та суміжних галузей.

За зміст наукових праць та достовірність наведених фактологічних і статистичних матеріалів відповідальність несуть автори публікацій та їхні наукові керівники. У збірнику зберігається стилістика та орфографія авторів матеріалів.

Склад організаційного комітету конференції

Керівництво оргкомітету

Мирослав Комар, д.т.н., професор, професор кафедри інформаційно-обчислювальних систем і управління

Христина Ліп'яніна-Гончаренко, д.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Надія Васильків, к.т.н., в.о. завідувача кафедри інформаційно-обчислювальних систем і управління

Члени програмного комітету

Василь Коваль, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління, заступник декана факультету комп'ютерних інформаційних технологій

Олександр Осолінський, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Павло Биковий, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Діана Загородня, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Ігор Майків, к.т.н., доцент кафедри інформаційно-обчислювальних систем і управління

Члени оргкомітету

Андрій Івасечко, викладач кафедри інформаційно-обчислювальних систем і управління

Дмитро Дюг, викладач кафедри інформаційно-обчислювальних систем і управління

Христина Юрків, студентка ФКІТ ЗУНУ, технік лабораторії з проблем інформаційних технологій

Мар'яна Соя, студентка ФКІТ ЗУНУ, лаборант кафедри інформаційно-обчислювальних систем і управління

Микола Телька, студент ФКІТ ЗУНУ, лаборант кафедри інформаційно-обчислювальних систем і управління, студентський декан факультету комп'ютерних інформаційних технологій

Мельник Назар
студент групи КН-42
nazarmelnik888@gmail.com
Осолінський Олександр
к.т.н., доцент
oso@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

АРХІТЕКТУРА СИСТЕМИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ НА БАЗІ ANDROID З ВИКОРИСТАННЯМ ІНТЕРНЕТУ РЕЧЕЙ

В сучасному світі зростає потреба у створенні комфортного, безпечного та енергоефективного житлового простору, що спонукає до активного розвитку розумних систем автоматизації будинку.

Також важливо зазначити, що домашня автоматизація може відігравати ключову роль у підтримці людей з особливими потребами, осіб похилого віку та людей з обмеженою мобільністю. Інтелектуальні системи можуть допомогти цим користувачам самостійно контролювати освітлення, температуру, двері чи вікна, створюючи безпечне і зручне середовище.

Попри всі переваги, існують деякі проблеми. Основні з них — це безпека та захист персональних даних. Відсутність єдиних стандартів, фрагментованість ринку, обмеження у пропускній здатності мереж, а також потреба в енергоефективності.

З технічного боку, апаратна частина таких систем зазвичай базується на доступних мікроконтролерах. [1]. [2].

Кожна система автоматизації має свою архітектуру, яка адаптується під конкретні завдання. Користувач у цьому випадку взаємодіє із системою через Android-додаток, який надсилає команди на контролер або отримує з нього дані про стан пристроїв [3, 4]. У деяких випадках використовуються розподілені системи, де частина логіки виконується безпосередньо на периферійних пристроях, що дозволяє підвищити надійність та швидкодію [2].

Метою подальшої роботи є побудова власної архітектури смарт-системи домашньої автоматизації на базі Android-застосунку, що взаємодіє з IoT-пристроями через протокол Wi-Fi, із можливістю хмарного збереження даних, моніторингу енергоспоживання, а також голосового управління.

Архітектура системи домашньої автоматизації

Для функціональної конфігурації HAS зазвичай достатньо традиційної мережі Wi-Fi та Ethernet. Система використовує різноманітні апаратні компоненти, включаючи плати Raspberry Pi 3, ESP 8266 і модулі Wi-Fi.

HAS API це простий HTTP API, який дозволяє дистанційно керувати основними портами обладнання (GPIO) і аналого-цифровими перетворювачами (ADC). Керування пристроєм включає загальний стан і налаштування пристрою, адміністрування портів включає конфігурацію та інформацію про порти, а зворотні виклики API включають виклики API, здійснені за допомогою

стандартів зворотних запитів HTTP, які надають широкий спектр функцій і варіантів використання, можуть здаватися дуже складними.

Сервер HAS служить центральним розташуванням, де розміщено зрозумілий онлайн-додаток. В результаті концентратор дозволить консолідувати адміністрування пристроїв, які використовує HAS (рисунок1).

Рисунок 1 – Топологія HAS

Концентратори керують користувачем під час спілкування з підключеними пристроями. Він також може працювати головним для інших підключених пристроїв за допомогою команди HAS. Головний пристрій керує підлеглими пристроями та надає їм доступ через виклики API.

Підлеглий пристрій може так само працювати як головний або головний для додаткових пристроїв. В результаті можна побудувати складні ланцюгові налаштування головного-підлеглого. Головний може перераховувати, додавати та видаляти підлеглі за допомогою викликів API, ексклюзивних для підлеглих. Головний пристрій розпізнає свої підлеглі (суб'єднані) пристрої за допомогою їхніх імен, тоді як головному потрібна додаткова інформація, щоб імена найслабших пристроїв могли бути змінені в будь-який час. Детальний дизайн показано на рисунку 2.

Одна з трьох ролей визначає рівень доступу до пристрою, а найважливіша роль адміністратора, має повний контроль над пристроєм і може змінювати конфігурацію та вигляд; звичайна роль, роль читання-запису, яка може читати з портів і записувати в них, може просто читати значення портів.

Рисунок 2 –Блок-схема зв'язку запропонованої HAS

Щоб полегшити автоматизацію, HAS дозволяє створювати правила, які визначають значення портів залежно від різних ситуацій. Вирази можна налаштувати на рівні пристрою або концентратора.

Висновки

Було розроблено архітектуру пропонованого рішення, яке базується на локальній обробці даних без обов'язкового підключення до хмари, що забезпечує більшу безпеку й автономність системи.

Запропонована архітектура є гнучкою, масштабованою та енергоефективною. Вона дозволяє автоматизувати управління побутовими приладами, оптимізувати витрати електроенергії, підвищити рівень безпеки та комфорту мешканців.

Список використаних джерел

1. Al Razib M., Javeed D., Khan M.T., Alkanhel R., Muthanna M.S.A. Cyber Threats Detection in Smart Environments Using SDN-Enabled DNN-LSTM Hybrid Framework // *IEEE Access*. – 2022. – Vol. 10. – P. 53015–53026.
2. Zamora-Izquierdo M.A., Santa J., Gomez-Skarmeta A.F. An Integral and Networked Home Automation Solution for Indoor Ambient Intelligence // *IEEE Pervasive Comput*. – 2010. – Vol. 9. – P. 66–77.
3. Debnath B., Dey R., Roy S. Smart switching system using bluetooth technology // Proceedings of the 2019 *Amity International Conference on Artificial Intelligence* (AICAI), Dubai, United Arab Emirates, 4–6 February 2019.
4. Kumar A. A Novel Decentralized Blockchain Architecture for the Preservation of Privacy and Data Security against Cyberattacks in Healthcare // *Sensors*. – 2022. – Vol. 22. – P. 5921.