Міністерство освіти і науки України
Херсонський державний університет

Кваліфікаційна наукова
праця на правах рукопису

Хоменко Євгеній Вікторович

УДК 004.738.5:004.7:004.42

ДИСЕРТАЦІЯ
Розроблення локальної IoT-системи моніторингу й контролю стану приміщень на базі платформи Home Assistant

121 Інженерія програмного забезпечення
(12 Інформаційні технології)

Подається на здобуття наукового ступеня
доктора філософії

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


Є.В. Хоменко

Науковий керівник: **Бабічев Сергій Анатолійович**,
доктор технічних наук, професор

Івано–Франківськ — 2026

АНОТАЦІЯ

Хоменко Є.В. Розроблення локальної IoT-системи моніторингу й контролю стану приміщень на базі платформи Home Assistant. Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 12.121 Інженерія програмного забезпечення. – Херсонський державний університет, Івано-Франківськ, 2026.

Загальна наукова проблема дослідження полягає у відсутності комплексного підходу до проєктування локальних автономних IoT-систем моніторингу та керування, які забезпечують повний контроль над даними, автономне виконання логіки автоматизацій, підвищений рівень безпеки та можливість подальшого розширення функціональності без залежності від зовнішніх хмарних сервісів.

Актуальність дослідження зумовлена стрімким поширенням IoT-рішень у сфері домашньої та офісної автоматизації, які здебільшого реалізуються у вигляді комерційних хмароорієнтованих екосистем. Такі рішення спрощують впровадження, однак водночас створюють обмеження щодо автономності, прозорості архітектури, безпеки та можливостей використання у дослідницьких і експериментальних сценаріях. У цьому контексті особливої ваги набуває розробка локальних серверних IoT-систем, орієнтованих на приватність, контрольованість та відтворюваність результатів.

Об'єктом дослідження є процеси функціонування IoT-систем моніторингу та керування станом приміщень як складних кіберфізичних систем, що забезпечують збір, передавання, обробку та використання сенсорних даних для автоматизованого керування параметрами середовища..

Предметом дослідження є архітектурні підходи, методи та моделі організації взаємодії між компонентами локальної автономної IoT-системи, спрямовані на забезпечення автономності її функціонування, підвищення рівня інформаційної безпеки та ефективного збору і локальної обробки сенсорних даних.

Метою роботи є розроблення та наукове обґрунтування архітектури і програмно-апаратного комплексу локальної автономної IoT-системи моніторингу й контролю стану приміщень, орієнтованої на інтеграцію гетерогенних сенсорних і виконавчих

пристроїв, підтримку різнорідних комунікаційних протоколів, локальну обробку та зберігання сенсорних даних, а також забезпечення підвищеного рівня інформаційної безпеки і автономного функціонування системи незалежно від зовнішніх хмарних сервісів.

Додатково мета роботи передбачає формування програмної інфраструктури для подальшого застосування аналітичних та інтелектуальних методів обробки даних у межах локальної обчислювальної платформи IoT-системи.

Для досягнення поставленої мети в дисертаційній роботі розв'язані такі основні завдання:

- виконати системний аналіз сучасних IoT-технологій, архітектурних підходів та програмних платформ з метою виявлення обмежень хмароорієнтованих рішень і обґрунтування доцільності застосування локальних автономних IoT-систем;
- дослідити та обґрунтувати вибір комунікаційних протоколів і стандартів IoT для локальних систем моніторингу з урахуванням вимог надійності, масштабованості, енергоефективності та сумісності пристроїв;
- розробити узагальнену багаторівневу архітектуру локальної IoT-системи, що забезпечує автономне функціонування, модульність, відмовостійкість, локальну обробку даних та підвищений рівень інформаційної безпеки;
- реалізувати програмно-апаратний прототип локальної IoT-системи з інтеграцією сенсорних і виконавчих пристроїв, сервісів збору та зберігання даних, а також механізмів автоматизації та керування;
- дослідити можливості локальної обробки та аналізу сенсорних даних у межах IoT-платформи на основі інтегрованого аналітичного контуру, що включає методи кластеризації, прогнозування та виявлення аномалій;
- оцінити функціональні, експлуатаційні та безпекові характеристики розробленої системи та експериментально підтвердити її ефективність порівняно з хмарними рішеннями за критеріями автономності, затримки обробки даних, рівня контролю над даними та масштабованості.

У першому розділі дисертації виконано огляд сучасного стану розвитку Internet of Things (IoT) та проаналізовано основні підходи до побудови систем моніторингу й автоматизації. Розглянуто еволюцію IoT-технологій, типові архітектурні моделі, стандарти та комунікаційні протоколи. Проаналізовано переваги та обмеження комерційних IoT-екосистем, а також можливості локальних серверних платформ.

Окрему увагу приділено питанням безпеки, автономності та контролю над даними. За результатами аналізу встановлено, що існуючі хмароорієнтовані рішення недостатньо придатні для дослідницьких і приватних IoT-систем, що дозволило сформувати дослідницьку прогалину, пов'язану з необхідністю локальних автономних IoT-рішень.

У другому розділі розроблено архітектуру локальної IoT-системи моніторингу та керування станом приміщень. Запропоновано багаторівневу структуру системи, яка охоплює рівень сенсорних і виконавчих пристроїв, комунікаційний рівень, рівень збору та зберігання даних, рівень керування автоматизаціями, а також підсистему безпеки. Обґрунтовано вибір протоколів взаємодії та принципів інтеграції компонентів з урахуванням вимог до автономної роботи, мінімальних затримок і локального виконання логіки керування.

У третьому розділі розглянуто питання аналізу даних, що формуються в процесі функціонування IoT-системи. Проаналізовано можливості використання аналітичних підходів для оцінювання стану приміщень, контролю параметрів мікроклімату, енергоспоживання та виявлення відхилень у роботі сенсорних вузлів. Показано доцільність локальної обробки даних у межах IoT-системи без залучення зовнішніх хмарних сервісів, що дозволяє підвищити автономність, зменшити затримки та забезпечити кращий контроль над даними.

У четвертому розділі реалізовано локальну IoT-систему моніторингу та керування на базі платформи Home Assistant. Виконано інтеграцію сенсорних і виконавчих пристроїв, налаштовано механізми збору даних і реалізовано сценарії автоматизації з локальним виконанням. Проведено експериментальне дослідження роботи системи в реальних умовах, оцінено її стабільність, автономність та зручність розширення. Отримані результати підтверджують ефективність використання локальної серверної платформи для побудови автономних IoT-систем моніторингу стану приміщень.

Наукова новизна одержаних результатів полягає у систематизації та обґрунтуванні архітектурних рішень для побудови локальних автономних IoT-систем моніторингу та контролю стану приміщень. У роботі сформовано цілісний підхід до організації взаємодії між сенсорними пристроями, комунікаційними протоколами та програмними компонентами локальної IoT-платформи з урахуванням вимог автономності, безпеки та керованості. Запропоновані рішення дозволяють узгодити роботу різнорідних пристроїв і сервісів у межах єдиної локальної

архітектури без залучення зовнішніх хмарних сервісів.

Практичне значення одержаних результатів полягає у можливості використання розроблених архітектурних підходів і програмної реалізації під час створення локальних IoT-систем автоматизації, моніторингу та керування інженерними параметрами приміщень. Отримані результати можуть бути застосовані для побудови приватних IoT-інсталяцій з підвищеним рівнем автономності, інформаційної безпеки та контролю над даними, а також використані як основа для подальших досліджень і експериментальних розробок у сфері локальних кіберфізичних систем.

Ключові слова: Інтернет речей (IoT); архітектура IoT-систем; периферійні обчислення (edge computing); бездротові сенсорні мережі; моніторинг; автоматизація; архітектура розподілених систем; проєктування програмних систем; моніторинг навколишнього середовища; інформаційна безпека; інформаційні системи; інтероперабельність систем; людино-комп'ютерна взаємодія; Home Assistant.

Список публікацій здобувача:

- *Список публікацій здобувача, у яких опубліковані основні наукові результати дисертації:*

1. Khomenko, Y., & Babichev, S. (2025). Modular IoT Architecture for Monitoring and Control of Office Environments Based on Home Assistant. IoT, 6(4), 69. https://doi.org/10.3390/iot6040069 (Scopus, WoS, quartile Q1)
2. Babichev, S., Yarema, O., Khomenko, Y., Senchyshen, D., & Durnyak, B. (2025). Sensor-Oriented Framework for Underwater Acoustic Signal Classification Using EMD–Wavelet Filtering and Bayesian-Optimized Random Forest. Sensors, 25(17), 5336. https://doi.org/10.3390/s25175336 (Scopus, WoS, quartile Q1)
3. Khomenko, Y., Babichev, S. (2025). Contemporary Methods, Models, and Software for Implementation and Optimization of IoT (Internet of Things) Systems: A Review. In: Babichev, S., Lytvynenko, V. (eds) Lecture Notes in Data Engineering, Computational Intelligence, and Decision-Making, Volume 2. ISDMCI 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 244. Springer, Cham , pp. 134–158. https://doi.org/10.1007/978-3-031-88483-2_7 (Scopus, book chapter, quartile Q4)

- *Список публікацій здобувача, які додатково відображають наукові результати дисертації:*

4. Хоменко Є.В., Бабічев С.А. (2025). Автономна IoT-система моніторингу мікроклімату аудиторій на основі відкритої DIY-архітектури. Том 8 № 1 (2025): Прикладні питання математичного моделювання, 2025, 245-254. https://doi.org/10.32782/mathematical-modelling/2025-8-1-24 (Категорія Б)
5. Хоменко Є. (2024).Сучасні методи, моделі та програмні засоби реалізації та оптимізації систем IoT (INTERNET OF THINGS),Збірник наукових праць "Information Technologies in Education"(ITE),(55), 72–84. https://doi.org/10.14308/ite000782 (Категорія Б)

- *Список публікацій здобувача, які засвідчують апробацію матеріалів дисертації:*

6. Khomenko Ye., Babichev S.(2024). Advantages and disadvantages of modern Internet of Things systems. Матеріали XX Міжнародної наукової інтернет-конференції „Intellectual Systems for Decision Making and Problems of Computational Intelligence", 20-23 червня 2024 року. С. 41-44. м. Усті на Лабі, Чехія
7. Khomenko Y.(2025). Comparison of commercial and local IoT systems for environmental monitoring in educational institutions. Abstracts of XIX International Scientific and Practical Conference. Krakow, Poland. Pp. 228-234.
8. Senchyshen D., Khomenko Y. (2025). Integration of IoT sensors into the KSU24 digital learning environment: infrastructure and pedagogical implications. Abstracts of XX International Scientific and Practical Conference. Plovdiv, Bulgaria. Pp. 222-226.
9. Khomenko Y. (2025) Modern voice assistants of intellectual systems and their use in Internet of Things (IoT) systems. Abstracts of XXII International Scientific and Practical Conference. Krakow, Poland. Pp. 208-214.
10. Хоменко Є., Лемещук О. (2023). Технічні аспекти та норми для систем керування «розумний будинок» в умовах війни, Матеріали IV International Scientific and Theoretical Conference „Features of the development of modern science in the pandemic's era,May 19, 2023. P. 73–75. Berlin, Germany.

ABSTRACT

Yevhenii V. Khomenko. Development of a local IoT system for monitoring and control of indoor environments based on the Home Assistant platform. Qualification scientific work, manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 12.121 Software Engineering. – Kherson State University, Ivano-Frankivsk, 2026.

The general scientific problem addressed in this dissertation lies in the lack of a comprehensive approach to the design of local autonomous IoT monitoring and control systems that ensure full control over data, autonomous execution of automation logic, an enhanced level of information security, and the possibility of further functional expansion without dependence on external cloud services.

The relevance of the research is determined by the rapid spread of IoT solutions in the field of home and office automation, which are predominantly implemented as commercial cloud-oriented ecosystems. Although such solutions simplify deployment, they simultaneously impose limitations on system autonomy, architectural transparency, information security, and applicability in research and experimental scenarios. In this context, the development of local server-based IoT systems focused on privacy, controllability, and reproducibility of results becomes particularly important.

The object of the study is the functioning processes of IoT systems for monitoring and controlling indoor environments as complex cyber-physical systems that ensure the collection, transmission, processing, and utilization of sensor data for automated management of environmental parameters.

The subject of the study is the architectural approaches, methods, and models for organizing interaction between the components of a local autonomous IoT system, aimed at ensuring its operational autonomy, increasing the level of information security, and achieving efficient collection and local processing of sensor data.

The goal of the work is the development and scientific substantiation of the architecture and a software-hardware complex for a local autonomous IoT system for monitoring and controlling indoor environments, oriented toward the integration of heterogeneous sensor and actuator devices, support for diverse communication protocols, local processing and storage of sensor data, as well as providing an

increased level of information security and autonomous system functioning independent of external cloud services.

Additionally, the goal of the work involves the formation of a software infrastructure for the further application of analytical and intelligent data processing methods within the local computing platform of the IoT system.

To achieve the stated objective of the dissertation, the following main tasks have been addressed:

- to perform a systematic analysis of modern IoT technologies, architectural approaches, and software platforms in order to identify the limitations of cloud-oriented solutions and substantiate the feasibility of using local autonomous IoT systems;
- to investigate and justify the selection of IoT communication protocols and standards for local monitoring systems, taking into account the requirements of reliability, scalability, energy efficiency, and device interoperability;
- to develop a generalized multi-layer architecture of a local IoT system that ensures autonomous operation, modularity, fault tolerance, local data processing, and an enhanced level of information security;
- to implement a hardware-software prototype of a local IoT system with the integration of sensor and actuator devices, data collection and storage services, as well as automation and control mechanisms;
- to investigate the capabilities of local processing and analysis of sensor data within the IoT platform based on an integrated analytical framework that includes clustering, prediction, and anomaly detection methods;
- to evaluate the functional, operational, and security characteristics of the developed system and to experimentally validate its effectiveness in comparison with cloud-based solutions according to the criteria of autonomy, data processing latency, data control, and scalability.

In the first chapter of the dissertation, the current state of development of Internet of Things (IoT) technologies is reviewed, and the main approaches to the construction of monitoring and automation systems are analyzed. The evolution of IoT technologies, typical architectural models, standards, and communication protocols is considered. The advantages and limitations of commercial IoT ecosystems, as well as the capabilities of local server-based platforms, are analyzed. Particular attention is paid to issues of information security, autonomy, and data control. Based on the results of the

analysis, it is established that existing cloud-oriented solutions are insufficiently suitable for research-oriented and private IoT systems, which makes it possible to identify a research gap related to the need for local autonomous IoT solutions.

In the second chapter, the architecture of a local IoT system for indoor monitoring and control is developed. A multi-layer system structure is proposed, covering the sensor and actuator layer, the communication layer, the data collection and storage layer, the automation control layer, and the security subsystem. The selection of interaction protocols and principles of component integration is substantiated, taking into account the requirements for autonomous operation, minimal latency, and local execution of control logic.

In the third chapter, the issues related to the analysis of data generated during the operation of the IoT system are considered. The possibilities of applying analytical approaches to assess indoor conditions, monitor microclimate parameters, energy consumption, and detect anomalies in the operation of sensor nodes are analyzed. The feasibility of local data processing within the IoT system without the involvement of external cloud services is demonstrated, which allows for increased system autonomy, reduced processing delays, and improved control over data storage and usage.

In the fourth chapter, a local IoT monitoring and control system based on the Home Assistant platform is implemented. Sensor and actuator devices are integrated, data collection mechanisms are configured, and automation scenarios with local execution are implemented. An experimental study of the system operation under real-world conditions is conducted, and its stability, autonomy, and extensibility are evaluated. The obtained results confirm the effectiveness of using a local server-based platform for building autonomous IoT systems for indoor condition monitoring.

The scientific novelty of the obtained results consists in the systematization and substantiation of architectural solutions for the development of local autonomous IoT systems for indoor monitoring and control. A comprehensive approach to organizing the interaction between sensor devices, communication protocols, and software components of a local IoT platform is formulated, taking into account the requirements for autonomy, information security, and controllability. The proposed solutions enable the integration of heterogeneous devices and services within a unified local architecture without involving external cloud services.

The practical significance of the obtained results lies in the possibility of applying the developed architectural approaches and software implementation in the

creation of local IoT systems for automation, monitoring, and control of indoor engineering parameters. The obtained results can be used to build private IoT installations with an increased level of autonomy, information security, and data control, as well as serve as a basis for further research and experimental developments in the field of local cyber-physical systems.

Key words: Internet of Things, IoT architecture, edge computing, wireless sensor networks, monitoring, automation, distributed system architecture, software system design, environmental monitoring, information security, information system, system interoperability, human-computer interaction, Home Assistant.

List of applicant's publications:

- *List of the applicant's publications in which the main scientific results of the dissertation are published:*

1. Khomenko, Y., & Babichev, S. (2025). Modular IoT architecture for monitoring and control of office environments based on Home Assistant. *IoT*, 6(4), 69. https://doi.org/10.3390/iot6040069 (Scopus, WoS, Q1)
2. Babichev, S., Yarema, O., Khomenko, Y., Senchyshen, D., & Durnyak, B. (2025). Sensor-oriented framework for underwater acoustic signal classification using EMD–wavelet filtering and Bayesian-optimized random forest. *Sensors*, 25(17), 5336. https://doi.org/10.3390/s25175336 (Scopus, WoS, Q1)
3. Khomenko, Y., & Babichev, S. (2025). Contemporary methods, models, and software for implementation and optimization of IoT (Internet of Things) systems: A review. In: Babichev, S., Lytvynenko, V. (Eds.), *Lecture Notes in Data Engineering, Computational Intelligence, and Decision-Making*, Volume 2. ISDMCI 2024. Lecture Notes on Data Engineering and Communications Technologies, vol. 244. Springer, Cham. https://doi.org/10.1007/978-3-031-88483-2_7 (Scopus, book chapter, Q4)

- *List of the applicant's publications additionally presenting the scientific results of the dissertation:*

4. Khomenko, Y. V., & Babichev, S. A. (2025). Autonomous IoT system for classroom microclimate monitoring based on an open DIY architecture. *Applied Issues of Mathematical Modelling*, 8(1). 245-254 https://doi.org/10.32782/mathematical-modelling/2025-8-1-24 (Ukrainian professional journal, Category B)
5. Khomenko, Y. (2025). Contemporary methods, models, and software tools for the implementation and optimization of IoT (Internet of Things) systems.

Information Technologies in Education, 55, 72–84. https://doi.org/10.14308/ite000782 (Ukrainian professional journal, Category B)

- *List of the applicant's publications confirming the approbation of the dissertation results:*

6. Khomenko, Y., & Babichev, S. (2024). Advantages and disadvantages of modern Internet of Things systems. In: *Proceedings of the 20th International Scientific Internet Conference "Intellectual Systems for Decision Making and Problems of Computational Intelligence"*, June 20–23, 2024, Ústí nad Labem, Czech Republic, pp. 41–44.
7. Khomenko, Y. (2025). Comparison of commercial and local IoT systems for environmental monitoring in educational institutions. In: *Abstracts of the XIX International Scientific and Practical Conference*, Krakow, Poland, pp. 228–234.
8. Senchyshen, D., & Khomenko, Y. (2025). Integration of IoT sensors into the KSU24 digital learning environment: Infrastructure and pedagogical implications. In: *Abstracts of the XX International Scientific and Practical Conference*, Plovdiv, Bulgaria, pp. 222–226.
9. Khomenko, Y. (2025). Modern voice assistants of intelligent systems and their use in Internet of Things (IoT) systems. In: *Abstracts of the XXII International Scientific and Practical Conference*, Krakow, Poland, pp. 208–214.
10. Khomenko, Y., & Lemeshchuk, O. (2023). Technical aspects and standards for smart home control systems under wartime conditions. In: *Proceedings of the IV International Scientific and Theoretical Conference "Features of the Development of Modern Science in the Pandemic's Era"*, May 19, 2023, Berlin, Germany, pp. 73–75.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IoT	Internet of Things (Інтернет речей)
CPS	Cyber-Physical System (кіберфізична система)
DIY	Do It Yourself (самостійна розробка)
API	Application Programming Interface (інтерфейс прикладного програмування)
REST	Representational State Transfer (архітектурний стиль взаємодії)
HTTP	Hypertext Transfer Protocol (протокол передавання гіпертексту)
WebSocket	Протокол двостороннього обміну даними через TCP-з'єднання
MQTT	Message Queuing Telemetry Transport (протокол обміну повідомленнями)
CoAP	Constrained Application Protocol (протокол для обмежених пристроїв)
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
LAN	Local Area Network (локальна обчислювальна мережа)
WAN	Wide Area Network (глобальна обчислювальна мережа)
VLAN	Virtual Local Area Network (віртуальна локальна мережа)
DNS	Domain Name System (система доменних імен)
IP	Internet Protocol
JSON	JavaScript Object Notation (формат обміну даними)
TLS	Transport Layer Security (протокол захисту транспортного рівня)
SSL	Secure Sockets Layer (криптографічний протокол захисту з'єднань)
ACL	Access Control List (список контролю доступу)
RBAC	Role-Based Access Control (рольова модель контролю доступу)
MFA	Multi-Factor Authentication (багатофакторна автентифікація)
OS	Operating System (операційна система)
HA	Home Assistant (локальна платформа автоматизації)
CO_2	Діоксид вуглецю
VOC	Volatile Organic Compounds (леткі органічні сполуки)
RH	Relative Humidity (відносна вологість повітря)

ВСТУП

Актуальність теми. На сучасному етапі розвитку інформаційних технологій концепція *Internet of Things* (IoT) набула широкого поширення у сфері моніторингу та керування станом приміщень, зокрема в системах житлової, офісної та комерційної автоматизації. Інтеграція сенсорних пристроїв, виконавчих модулів і програмних сервісів у межах єдиної інформаційно-комунікаційної інфраструктури забезпечує можливість безперервного контролю параметрів середовища, таких як температура, вологість, освітленість, рівень енергоспоживання та показники безпеки, а також реалізації складних автоматизованих сценаріїв керування. У результаті IoT-системи поступово еволюціонують від допоміжних засобів автоматизації до складних кіберфізичних систем, що безпосередньо впливають на комфорт, енергоефективність і безпеку експлуатації приміщень.

Водночас стрімке зростання кількості IoT-пристроїв, а також різноманіття використовуваних комунікаційних протоколів і програмних платформ зумовлює суттєве ускладнення архітектури таких систем. Сучасні IoT-рішення характеризуються високим рівнем гетерогенності апаратних компонентів, мультипротокольною взаємодією, розподіленою логікою керування та необхідністю обробки значних обсягів сенсорних даних у режимі реального часу. За таких умов особливої актуальності набувають питання архітектурної організації IoT-систем, забезпечення їхньої надійності, масштабованості, відмовостійкості та передбачуваної поведінки в процесі експлуатації.

Переважна більшість сучасних комерційних IoT-екосистем орієнтована на використання хмароорієнтованих архітектур, у межах яких зберігання даних, виконання автоматизацій і керування пристроями здійснюються із залученням зовнішніх сервісів. Такий підхід спрощує процес розгортання систем і полегшує інтеграцію з мобільними застосунками та віддаленими сервісами, однак водночас зумовлює низку суттєвих обмежень. До них належать залежність функціонування системи від стабільності інтернет-з'єднання, зниження рівня автономності, обмежений контроль над процесами обробки та зберігання даних, а також підвищені ризики порушення інформаційної безпеки й втрати приватності. Для систем моніторингу стану приміщень зазначені недоліки є критичними, оскільки такі си-

стеми повинні забезпечувати безперервну, стабільну та надійну роботу незалежно від доступності зовнішньої мережевої інфраструктури.

У зв'язку з цим зростає актуальність наукових досліджень, спрямованих на розробку локальних автономних IoT-систем, у яких ключові функції збору, обробки й аналізу даних, а також логіка автоматизованого керування реалізуються в межах локальної обчислювальної інфраструктури. Локальний підхід дає змогу зменшити затримки обробки подій, підвищити стійкість системи до мережевих збоїв, забезпечити повний контроль над життєвим циклом даних і реалізувати гнучкі та прозорі механізми захисту інформації. Разом із тим проєктування таких систем потребує науково обґрунтованого вибору архітектурних рішень, узгодження різнорідних комунікаційних протоколів і програмних компонентів, а також забезпечення ефективної взаємодії між фізичним, комунікаційним, серверним і прикладним рівнями системи.

Аналіз сучасних наукових публікацій і практичних реалізацій свідчить про те, що значна частина досліджень зосереджена або на окремих протокольних аспектах IoT, або на застосуванні готових комерційних платформ автоматизації. Водночас питання комплексного архітектурного проєктування локальних автономних IoT-систем моніторингу стану приміщень, орієнтованих на дослідницькі, експериментальні та прикладні сценарії, залишаються недостатньо систематизованими. Зокрема, подальшого опрацювання потребують підходи до інтеграції пристроїв різних типів і виробників, організації локального серверного рівня, забезпечення інформаційної безпеки та централізованої керованості IoT-системи в цілому.

Таким чином, актуальність дисертаційної роботи зумовлена необхідністю розробки та наукового обґрунтування підходу до проєктування і дослідження локальної IoT-системи моніторингу й контролю стану приміщень, яка поєднує автономність, архітектурну гнучкість і підвищений рівень контролю над даними та процесами керування. Розв'язання зазначених завдань є важливим як з теоретичної, так і з практичної точки зору та відповідає сучасним тенденціям розвитку IoT-технологій і кіберфізичних систем.

Мета й завдання дослідження. Метою дисертаційної роботи є розроблення та наукове обґрунтування архітектури і програмно-апаратного комплексу локальної автономної IoT-системи моніторингу й контролю стану приміщень, орієнтованої на інтеграцію гетерогенних сенсорних і виконавчих пристроїв, підтримку

різнорідних комунікаційних протоколів, локальну обробку та зберігання сенсорних даних, а також забезпечення підвищеного рівня інформаційної безпеки і автономного функціонування системи незалежно від зовнішніх хмарних сервісів.

Додатково мета роботи передбачає формування програмної інфраструктури для подальшого застосування аналітичних та інтелектуальних методів обробки даних у межах локальної обчислювальної платформи IoT-системи.

Для досягнення поставленої мети в дисертаційній роботі необхідно розв'язати такі основні завдання:

- виконати системний аналіз сучасних IoT-технологій, архітектурних підходів та програмних платформ з метою виявлення обмежень хмароорієнтованих рішень і обґрунтування доцільності застосування локальних автономних IoT-систем;
- дослідити та обґрунтувати вибір комунікаційних протоколів і стандартів IoT для локальних систем моніторингу з урахуванням вимог надійності, масштабованості, енергоефективності та сумісності пристроїв;
- розробити узагальнену багаторівневу архітектуру локальної IoT-системи, що забезпечує автономне функціонування, модульність, відмовостійкість, локальну обробку даних та підвищений рівень інформаційної безпеки;
- реалізувати програмно-апаратний прототип локальної IoT-системи з інтеграцією сенсорних і виконавчих пристроїв, сервісів збору та зберігання даних, а також механізмів автоматизації та керування;
- дослідити можливості локальної обробки та аналізу сенсорних даних у межах IoT-платформи на основі інтегрованого аналітичного контуру, що включає методи кластеризації, прогнозування та виявлення аномалій;
- оцінити функціональні, експлуатаційні та безпекові характеристики розробленої системи та експериментально підтвердити її ефективність порівняно з хмарними рішеннями за критеріями автономності, затримки обробки даних, рівня контролю над даними та масштабованості.

Об'єкт та предмет дослідження. Об'єктом дослідження є процеси функціонування IoT-систем моніторингу та керування станом приміщень як складних кіберфізичних систем, що забезпечують збір, передавання, обробку та використання сенсорних даних для автоматизованого керування параметрами середовища.

Предметом дослідження є архітектурні підходи, методи та моделі організації взаємодії між компонентами локальної автономної IoT-системи, спрямовані на

забезпечення автономності її функціонування, підвищення рівня інформаційної безпеки та ефективного збору і локальної обробки сенсорних даних.

Методи дослідження. Для досягнення поставленої мети та розв'язання визначених завдань у дисертаційній роботі використано сукупність загальнонаукових і спеціалізованих методів дослідження, зокрема:

- методи аналізу, систематизації та узагальнення застосовано для дослідження сучасних наукових публікацій, стандартів і практичних реалізацій у сфері *Internet of Things*, а також для проведення порівняльного аналізу комерційних IoT-екосистем і локальних серверних платформ автоматизації;
- методи системного аналізу та структурного моделювання використано для розроблення архітектури локальної автономної IoT-системи, визначення її функціональних рівнів, характеру взаємодії між компонентами та обґрунтування вибору комунікаційних протоколів;
- методи проєктування програмних систем і компонентно-орієнтованого підходу застосовано для формалізації логіки взаємодії між IoT-пристроями, сервісами збору й обробки даних та керуючими модулями в межах локальної IoT-платформи;
- експериментальні методи дослідження використано під час реалізації та аналізу прототипу локальної IoT-системи на базі платформи *Home Assistant*, а також для оцінювання її функціональних характеристик, автономності функціонування та стійкості до мережевих збоїв.

Наукова новизна отриманих результатів. Основними науковими результатами, що визначають новизну дисертаційної роботи та виносяться на захист, є такі положення:

1. **Вперше** систематизовано та науково обґрунтовано архітектурний підхід до побудови локальних автономних IoT-систем моніторингу та керування станом приміщень, який забезпечує повну незалежність функціонування системи від зовнішніх хмарних сервісів, локальне виконання автоматизаційних сценаріїв і контроль життєвого циклу сенсорних даних у межах локальної обчислювальної інфраструктури.
2. **Вперше** запропоновано багаторівневу архітектурну модель локальної IoT-системи моніторингу та контролю стану приміщень, що включає фізичний, комунікаційний, інформаційний, керуючий, безпековий, сервісний та пре-

зентаційний рівні, забезпечує функціональне розмежування компонентів системи та створює передумови для її масштабування й адаптації.

3. **Удосконалено** метод інтеграції гетерогенних IoT-пристроїв і комунікаційних протоколів у межах локальної обчислювальної платформи, що забезпечує їх узгоджену взаємодію та уніфіковані механізми доступу до сенсорних даних і керування виконавчими пристроями.
4. **Удосконалено** комплексний архітектурний підхід до забезпечення інформаційної безпеки локальних IoT-систем, який поєднує мережеву сегментацію, шифрування транспортного рівня, механізми контролю доступу, журналювання подій та багатофакторну автентифікацію.
5. **Набуло подальшого розвитку** формування сервісного рівня локальної IoT-системи у вигляді уніфікованого програмного інтерфейсу (API) з підтримкою протоколів REST та WebSocket, що забезпечує асинхронну взаємодію компонентів системи в режимі реального часу та можливість підключення зовнішніх програмних і аналітичних модулів.
6. **Набуло подальшого розвитку** застосування підходів локальної (edge) обробки сенсорних даних у системах моніторингу приміщень, що дозволяє зменшити затримки обробки подій, підвищити стійкість системи до мережевих збоїв та забезпечити збереження конфіденційності даних у межах локальної інфраструктури.

Практичне значення отриманих результатів. Практичне значення результатів дисертаційної роботи полягає у можливості їх використання під час проєктування, розгортання та експлуатації локальних автономних IoT-систем моніторингу й керування станом приміщень. Запропоновані архітектурні рішення та підходи можуть бути застосовані при створенні систем автоматизації житлових, офісних і виробничих об'єктів, для яких є критичними вимоги автономності функціонування, підвищеного рівня інформаційної безпеки та контролю над процесами збору й обробки сенсорних даних.

Розроблена багаторівнева архітектура локальної IoT-системи сприяє впорядкуванню процесів інтеграції сенсорних і виконавчих пристроїв, спрощує масштабування системи та підвищує надійність її функціонування в умовах обмеженого або нестабільного мережевого з'єднання. Запропоновані підходи до організації комунікаційної взаємодії, формування сервісного рівня та керування автоматизованими сценаріями можуть використовуватися як методична основа для розроблення

подібних систем на різних програмно-апаратних платформах.

Практичну цінність мають також запропоновані архітектурні механізми забезпечення інформаційної безпеки локальних IoT-систем, зокрема використання мережевої сегментації, криптографічного захисту каналів зв'язку та багаторівневого контролю доступу. Зазначені рішення можуть бути використані при впровадженні IoT-рішень у середовищах з підвищеними вимогами до захисту інформації та збереження конфіденційності даних.

Результати дисертаційної роботи можуть бути використані у науково-дослідній та освітній діяльності як приклад побудови локальної автономної IoT-інфраструктури, а також як експериментальна платформа для дослідження методів збору й аналізу сенсорних даних, автоматизації та керування в кіберфізичних системах. Окремі положення роботи доцільно використовувати під час підготовки навчальних матеріалів і викладання дисциплін, пов'язаних з *Internet of Things*, архітектурою програмних систем та розподіленими системами керування.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційну роботу виконано відповідно до пріоритетних напрямів наукової діяльності Херсонського державного університету, що охоплюють дослідження актуальних проблем фізико-математичних і технічних наук. Зокрема, дослідження відповідає науковому напряму, пов'язаному з розвитком інтелектуальних інформаційних та інформаційно-аналітичних технологій, інтегрованих систем баз даних і знань, а також технологій і засобів проєктування програмних продуктів та інформаційних систем.

Тематика дисертаційної роботи узгоджується з науковими дослідженнями у галузі побудови розподілених і локальних інформаційних систем, застосування сучасних IoT-технологій, архітектурних підходів до організації кіберфізичних систем, а також методів забезпечення їх автономності, надійності та інформаційної безпеки. Розробка локальної IoT-системи моніторингу й контролю стану приміщень безпосередньо пов'язана з напрямами інтеграції гетерогенних джерел даних, обробки сенсорної інформації та створення програмно-апаратних комплексів, орієнтованих на автономне функціонування в межах локальної інфраструктури.

Результати дисертаційного дослідження також корелюють із науковими роботами, спрямованими на розвиток методів системного аналізу, математичного моделювання та оптимізації складних технічних об'єктів і процесів. Запропоновані в роботі архітектурні рішення та підходи до організації локальної IoT-платформи

можуть бути використані як методологічна та технологічна основа для подальших досліджень у сфері проєктування інтелектуальних інформаційних систем, систем автоматизації та моніторингу, а також при створенні експериментальних і дослідницьких IoT-інсталяцій.

Таким чином, дисертаційна робота органічно вписується в загальну тематику наукових досліджень університету, відповідає його науковим програмам і планам та узгоджується з актуальними науково-технічними напрямами розвитку інформаційних і програмних технологій.

Апробація результатів дисертації. Основні положення, проміжні висновки та результати дисертаційного дослідження пройшли апробацію та були представлені у вигляді доповідей і наукових публікацій на міжнародних наукових конференціях і науково-практичних форумах. Зокрема, результати роботи доповідалися на XX Міжнародній науковій інтернет-конференції «Intellectual Systems for Decision Making and Problems of Computational Intelligence» (20–23 червня 2024 р., Усті-над-Лабем, Чехія), IV International Scientific and Theoretical Conference «Features of the Development of Modern Science in the Pandemic's Era» (19 травня 2023 р., Берлін, Німеччина), а також на XIX, XX та XXII International Scientific and Practical Conferences, що відбулися відповідно у Кракові (Польща) та Пловдиві (Болгарія).

Представлені на конференціях результати стосувалися питань архітектури IoT-систем, порівняльного аналізу комерційних і локальних рішень, інтеграції сенсорних пристроїв і комунікаційних протоколів, а також застосування IoT-технологій в освітніх та інтелектуальних середовищах.

Публікації. Основні результати дисертаційної роботи викладено у 10 наукових публікаціях, серед яких статті у міжнародних рецензованих наукових журналах і колективних монографіях. Частина робіт опублікована у виданнях, що індексуються в міжнародних наукометричних базах даних *Scopus* та *Web of Science* (WoS), зокрема у журналах *Sensors* (квартиль Q1), *IoT* (квартиль Q1), а також у серії *Lecture Notes on Data Engineering and Communications Technologies* видавництва Springer (розділ у колективній монографії, квартиль Q4).

Крім того, за темою дисертації опубліковано дві статті у наукових виданнях України категорії Б та п'ять публікацій у вигляді тез доповідей на міжнародних наукових конференціях. Усі опубліковані праці безпосередньо відображають основні положення, наукові результати та висновки дисертаційного дослідження.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, основної частини, що містить чотири розділи, загальних висновків, списку використаних джерел та додатків. Повний обсяг дисертаційної роботи становить 207 сторінок, з яких 138 сторінок припадає на основний текст; робота містить 15 таблиць, 69 рисунків, 12 формул та 6 додатів.

У першому розділі здійснено огляд сучасного стану розвитку технологій *Internet of Things* та проаналізовано основні підходи до побудови систем моніторингу й автоматизації. Розглянуто еволюцію IoT-технологій, типові архітектурні моделі, стандарти та комунікаційні протоколи. Проведено аналіз комерційних IoT-екосистем і локальних серверних платформ, а також їх переваг і обмежень з точки зору автономності функціонування, інформаційної безпеки та контролю над даними. Окрему увагу приділено питанням кібербезпеки IoT-систем. За результатами розділу сформовано науково обґрунтовану дослідницьку прогалину, пов'язану з недостатньою опрацьованістю локальних автономних IoT-рішень.

У другому розділі розроблено архітектуру локальної IoT-системи моніторингу та керування станом приміщень. Запропоновано багаторівневу структуру системи, що охоплює рівень сенсорних і виконавчих пристроїв, комунікаційний рівень, рівень збору та зберігання даних, рівень керування автоматизаціями, сервісний рівень і підсистему безпеки. Обґрунтовано вибір протоколів взаємодії та принципів інтеграції компонентів з урахуванням вимог до автономної роботи, мінімальних затримок і локального виконання логіки керування.

У третьому розділі розглянуто питання аналізу даних, що формуються в процесі функціонування локальної IoT-системи. Проаналізовано підходи до оцінювання стану приміщень, контролю параметрів мікроклімату, енергоспоживання та виявлення відхилень у роботі сенсорних вузлів. Обґрунтовано доцільність локальної обробки даних у межах IoT-системи без залучення зовнішніх хмарних сервісів, що дозволяє підвищити автономність системи, зменшити затримки обробки подій та забезпечити контроль над збереженням і використанням даних.

У четвертому розділі виконано практичну реалізацію локальної IoT-системи моніторингу та керування станом приміщень на базі платформи *Home Assistant*. Проведено інтеграцію сенсорних і виконавчих пристроїв, налаштовано механізми збору даних і реалізовано сценарії автоматизації з локальним виконанням. Здійснено експериментальне дослідження роботи системи в реальних умовах, оцінено її стабільність, автономність, рівень інформаційної безпеки та можливості

подальшого розширення. Отримані результати підтверджують ефективність використання локальної серверної платформи для побудови автономних IoT-систем моніторингу стану приміщень.

РОЗДІЛ 1
СУЧАСНІ МЕТОДИ, МОДЕЛІ ТА ПРОГРАМНІ ЗАСОБИ ВПРОВАДЖЕННЯ ТА ОПТИМІЗАЦІЇ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ (IOT)

Аналіз сучасних підходів до реалізації та оптимізації систем Інтернету речей (IoT), проведений у вступній частині дисертації, продемонстрував стрімкий розвиток технологій у сфері автоматизації та повсякденного застосування "розумних"рішень. Більшість існуючих платформ орієнтовані на використання хмарних сервісів, що забезпечує зручність інтеграції та швидкість розгортання. Проте така модель супроводжується низкою суттєвих обмежень, зокрема підвищеними ризиками для безпеки даних, залежністю від стабільності зовнішніх мереж та обмеженим контролем над інфраструктурою.

В умовах зростання вимог до приватності, автономності та надійності IoT-систем актуалізується необхідність у створенні локальних архітектур, здатних забезпечувати повний контроль користувача над середовищем, незалежність від зовнішніх ресурсів та високу адаптивність до змінних умов експлуатації. Розробка таких систем потребує інтеграції гнучких методів, відкритих протоколів і програмних платформ, які б одночасно забезпечували масштабованість, безпеку та простоту використання.

У цьому контексті особливу увагу привертають рішення, що дозволяють організувати локальне зберігання даних, гнучке керування пристроями різних виробників та створення сценаріїв автоматизації без необхідності підключення до хмарних сервісів. Серед відкритих і доступних рішень платформи на кшталт Home Assistant виділяються своєю технічною зрілістю, широким спектром підтримуваних інтеграцій та високим рівнем безпеки.

У першому розділі дисертації буде представлено огляд сучасних методів, моделей і програмних засобів реалізації та оптимізації систем IoT, із фокусом на їх здатність забезпечити автономність, безпеку, масштабованість і підтримку мультипротокольної взаємодії в локальному середовищі. Проведений аналіз стане основою для формулювання вимог до архітектури запропонованої системи та вибору відповідних технологічних рішень.

Основні результати, представлені у цьому розділі, опубліковані у роботах автора [1–4]

1.1. Актуальність дослідження та аналіз проблем побудови IoT-систем

Технології Інтернету речей (Internet of Things, IoT) набули широкого застосування у сучасних кіберфізичних системах і відіграють ключову роль у розвитку цифрової інфраструктури [5–7]. Під IoT зазвичай розуміють сукупність взаємопов'язаних пристроїв, здатних здійснювати вимірювання параметрів середовища, обмінюватися даними та виконувати керуючі дії у фізичному просторі [8]. Завдяки цьому IoT виступає базовою технологією для автоматизації житлових, офісних і міських середовищ, забезпечуючи моніторинг стану приміщень, оптимізацію енергоспоживання та підвищення рівня комфорту користувачів [9, 10]. У межах цифрової трансформації саме IoT створює технічні передумови для реалізації концепцій Smart Home, Smart Office та Smart City, у яких автоматизуються підсистеми освітлення, вентиляції, клімат-контролю, безпеки та керування доступом.

Водночас розвиток IoT-систем супроводжується низкою науково-технічних проблем, які безпосередньо впливають на вибір архітектурних рішень та підходів до інтеграції пристроїв. До найбільш суттєвих викликів належать:

- масштабованість при зростанні кількості сенсорів, актуаторів і вузлів керування;
- забезпечення безпеки та конфіденційності даних;
- складність взаємодії між різними протоколами та пристроями різних виробників;
- енергоспоживання, що є критичним фактором для автономних сенсорних систем;
- залежність від хмарних сервісів у більшості комерційних рішень;
- недостатня інтероперабельність, спричинена фрагментацією стандартів.

Значна частина сучасних IoT-рішень побудована за хмароцентричною моделлю, у межах якої зберігання даних і обробка подій здійснюються на віддалених серверах [7, 9–11]. Такий підхід спрощує початкове розгортання та масштабування систем, проте супроводжується низкою обмежень, зокрема збільшенням затримок, залежністю від стабільності інтернет-з'єднання, ризиками витоку даних,

обмеженим контролем з боку користувача та наявністю єдиних точок відмови [8, 12]. У практичних інсталяціях, особливо в комерційних та офісних середовищах, зазначені недоліки набувають критичного характеру, що підтверджується результатами низки досліджень [13–15].

У період 2020-2025 рр. у наукових і прикладних роботах спостерігається стійка тенденція переходу до локальних IoT-рішень, орієнтованих на використання власних серверів або edge-пристроїв. Перенесення обробки даних на локальний рівень дозволяє зменшити затримку реакцій, підвищити надійність системи, зберегти конфіденційність даних і спростити інтеграцію гетерогенних пристроїв. Дослідження [13, 16] демонструють, що локальні контролери здатні ефективно виконувати частину аналітики та автоматизацій без залучення хмарних сервісів. Крім того, у роботах [17, 18] показано, що сучасні апаратні платформи забезпечують можливість реалізації локальних алгоритмів машинного навчання, що додатково підсилює тренд на автономність IoT-систем.

У цьому контексті особливого значення набувають універсальні локальні платформи, які підтримують розгортання без обов'язкової прив'язки до хмарних сервісів. До таких платформ належать *Home Assistant* [19–25], *openHAB* [26, 27], *Domoticz* [28, 29] та інші. Серед них платформа *Home Assistant* вирізняється широкою екосистемою інтеграцій, підтримкою відкритих протоколів та можливістю повністю локального розгортання, що робить її одним із найбільш придатних рішень для побудови автономних IoT-систем.

Попри суттєвий прогрес у розвитку локальних IoT-платформ, аналіз сучасної літератури свідчить про відсутність формалізованої архітектурної моделі, яка б уніфіковано поєднувала:

- автономність від зовнішніх сервісів;
- підтримку різних протоколів і типів пристроїв;
- можливість інтеграції правило-орієнтованої логіки та методів машинного навчання;
- структурований поділ функцій за рівнями;
- експериментально підтверджені показники надійності та продуктивності.

Необхідність усунення зазначених обмежень і формування цілісного архітектурного підходу до побудови локальних IoT-систем визначає актуальність цього дослідження. Подальші розділи роботи присвячені аналізу архітектурних моделей IoT, обґрунтуванню вимог до локальних систем автоматизації та розгляду

підходів до їх практичної реалізації.

1.1.1. Сучасний стан і напрями розвитку IoT-систем.

Формування та розвиток концепції Internet of Things (IoT) відбувалися поступово й перебували у тісному взаємозв'язку з еволюцією обчислювальної техніки, телекомунікаційних технологій та вбудованих систем [30]. За оцінками компанії Cisco, масовий перехід до IoT-моделі можна віднести до 2008-2009 років, коли кількість пристроїв, підключених до мережі Інтернет, уперше перевищила чисельність населення планети [9]. У 2015 році їх кількість оцінювалася більш ніж у 15 млрд, а до 2020 року прогнозувалося зростання понад 30 млрд підключених пристроїв [10, 19]. Це засвідчило перехід IoT від експериментальної концепції до одного з ключових елементів цифрової економіки.

Сучасне розуміння IoT сформувалося внаслідок розвитку концепцій Smart Home та pervasive computing. Інтелектуальні житлові середовища стали першими масштабними платформами для впровадження сенсорних мереж, автоматизованого керування та віддаленого моніторингу, що сприяло підвищенню комфорту, безпеки та енергоефективності житлових і комерційних приміщень [28, 31]. Надалі зазначені підходи були поширені на промислові, медичні та сервісні системи, а також на міські інфраструктури в межах концепції Smart City [26].

Перші прообрази IoT-систем з'явилися задовго до формального введення відповідного терміна. Одним із ключових ранніх проєктів вважається *ECHO IV* (1960 р.), у межах якого було реалізовано автоматизоване керування окремими побутовими пристроями. Подальшим важливим кроком стало створення у 1975 році стандарту *X-10* - першої технології передавання керуючих сигналів електромережею будинку, що заклала основу централізованого керування побутовими електроприладами [21]. У 1973 році також було запатентовано перший RFID-тег, який започаткував розвиток безконтактної ідентифікації та простежуваності об'єктів.

Наступний етап розвитку пов'язаний із підключенням побутових і промислових пристроїв до глобальної мережі. Підключення торгового автомата в Carnegie Mellon University (1982 р.), інтернет-тостера на конференції Interop '89 (1989 р.) та кавоварки у Кембриджському університеті (1993 р.) продемонстрували можливості віддаленого моніторингу у режимі реального часу. Важливим технологічним зрушенням стало впровадження Ethernet-пристроїв, зокрема принтера HP LaserJet IIISi (1991 р.), що сприяло стандартизації мережевої взаємодії периферійних пристроїв.

Початок 2000-х років відзначився переходом до енергоефективних бездротових технологій, придатних для масового розгортання сенсорних мереж. Представлення протоколу *Zigbee* у 2003 році забезпечило технічні можливості для створення масштабованих мереж з великою кількістю вузлів і низьким енергоспоживанням, що стало критичним для побутових і офісних IoT-інсталяцій. У 2010 році було стандартизовано *Bluetooth Low Energy (BLE)*, який став основою для портативних і батарейних IoT-пристроїв, трекерів та маячків.

Подальший розвиток IoT-технологій був спрямований на підвищення інтероперабельності та зменшення фрагментації ринку. Поява протоколу *iBeacon* у 2014 році відкрила можливості для контекстної взаємодії у внутрішніх просторах, зокрема для навігації та реалізації сценаріїв присутності. Представлення відкритого стандарту *Matter* у 2022 році стало відповіддю на проблему несумісності пристроїв різних виробників і спробою уніфікувати модель взаємодії розумних пристроїв. На рис. 1.1 наведено узагальнену часову шкалу розвитку IoT-технологій від 1960-х років до 2022 року, яка демонструє найінтенсивніший період розвитку у 2000-2010 роках [21].

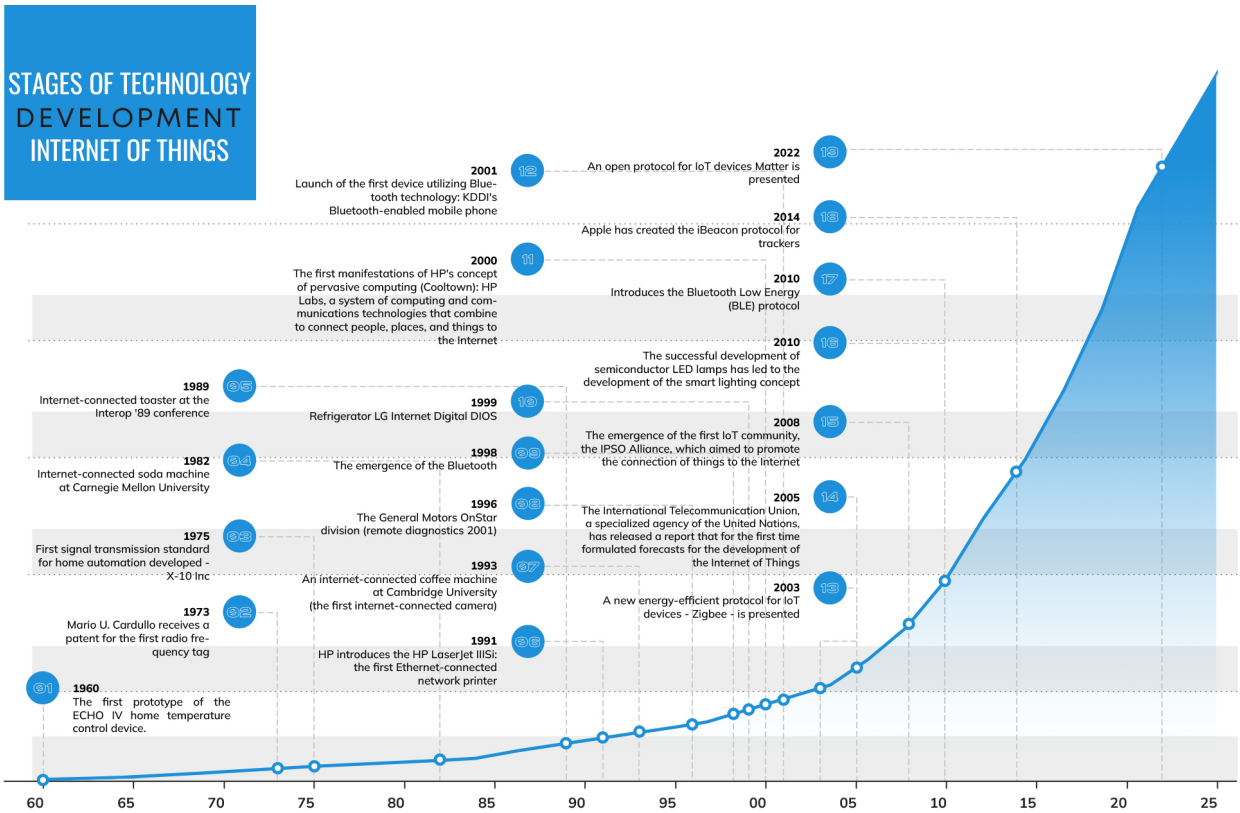

Рис. 1.1: Основні етапи розвитку IoT-технологій

На основі аналізу джерел можна виділити такі ключові етапи розвитку IoT [22]:

- 1960 р. - *ECHO IV*, перший прототип автоматизованого керування домашніми пристроями;
- 1973 р. - патентування RFID-тега;
- 1975 р. - поява стандарту *X-10*;
- 1982 р. - підключення торгового автомата до Інтернету (CMU);
- 1989 р. - демонстрація інтернет-тостера;
- 1991 р. - перший Ethernet-принтер HP LaserJet IIISi;
- 1993 р. - перша інтернет-камера (кавоварка у Cambridge University);
- 1996 р. - запуск сервісу *OnStar*;
- 1998 р. - створення *Bluetooth SIG*;
- 1999 р. - випуск підключеного холодильника LG DIOS;
- 2000 р. - концепція *Cooltown* (pervasive computing);
- 2001 р. - перші комерційні Bluetooth-пристрої;
- 2003 р. - представлення протоколу *Zigbee*;
- 2005 р. - звіт ITU щодо перспектив розвитку IoT;
- 2008 р. - заснування *IPSO Alliance*;
- 2010 р. - поява BLE та концепції smart lighting;
- 2014 р. - впровадження *iBeacon*;
- 2022 р. - представлення відкритого стандарту *Matter*.

Узагальнюючи наведені етапи, можна зазначити, що розвиток IoT-технологій від перших експериментальних рішень до спеціалізованих бездротових протоколів і відкритих стандартів поступово сформував технологічну основу для сучасних систем автоматизації. Послідовне ускладнення функціональності пристроїв, перехід до мережевої взаємодії та поява енергоефективних протоколів зв'язку зумовили сучасні вимоги до масштабованості, безпеки та інтероперабельності IoT-рішень [30, 32, 33]. Зазначені історичні передумови надалі враховуються під час аналізу архітектурних моделей і структур побудови IoT-систем.

1.1.2. Класифікація сучасних IoT-архітектур (моделі та структури).

У науковій літературі [25, 34, 35] архітектури Internet of Things (IoT) зазвичай розглядаються як багаторівневі моделі, призначені для впорядкування функцій збору, передавання, обробки та використання даних. Такі моделі виконують подвійну роль: з одного боку, вони узагальнюють типові підходи до побудови кіберфізичних систем, а з іншого - слугують концептуальною основою для розроблен-

ня конкретних апаратно-програмних рішень у дослідницьких і комерційних платформах. Попри відмінності у кількості рівнів та їх функціональному наповненні, більшість архітектурних моделей орієнтовані на забезпечення інтероперабельності, керованості, масштабованості та безпеки IoT-систем.

1.1.2.1. Трирівнева архітектура IoT.

Трирівнева модель є одним із перших формалізованих описів IoT-архітектури та широко використовується як базовий рівень абстракції. Рівень *Perception* охоплює сенсорні вузли, виконавчі механізми та вбудовані мікроконтролери, що здійснюють первинне вимірювання фізичних параметрів середовища та генерацію подій. Рівень *Network* забезпечує передавання даних за допомогою дротових і бездротових протоколів, маршрутизацію пакетів та базову комунікацію між вузлами системи. Рівень *Application* реалізує прикладні сервіси, аналітичні функції та користувацькі інтерфейси, призначені для взаємодії з оператором або кінцевим користувачем.

Попри свою простоту та зручність для концептуального опису, трирівнева модель не охоплює низку критично важливих аспектів сучасних IoT-систем, зокрема механізми обробки даних на периферії мережі, стандартизовані підходи до забезпечення безпеки, керування життєвим циклом пристроїв та підтримку масштабного мультипротокольного середовища. У зв'язку з цим у наукових роботах вона зазвичай використовується як початковий рівень абстракції, який надалі уточнюється або розширюється.

1.1.2.2. П'ятирівнева архітектура IoT.

П'ятирівнева модель розширює трирівневу структуру шляхом введення двох додаткових рівнів, що відображають зростаючі вимоги до систем обробки та управління даними. Рівень *Processing* включає операції фільтрації, агрегації, класифікації та короткочасного зберігання даних, що дозволяє розвантажити прикладні сервіси та забезпечити обробку подій у напівреальному часі. Рівень *Business* представляє стратегічний рівень прийняття рішень, аналітики, інтеграції з корпоративними інформаційними системами та реалізації політик управління.

П'ятирівнева архітектура є актуальною для промислових і комерційних IoT-рішень, однак залишає відкритими питання формалізації механізмів безпеки, централізованого керування пристроями та підтримки гетерогенних протоколів, що є особливо критичним для локальних автономних систем.

1.1.2.3. Семирівнева архітектура Cisco IoT.

Семирівнева модель Cisco є однією з найбільш цитованих і детально структурованих архітектур IoT. У межах цієї моделі виокремлено такі рівні: пристроїв, підключення, периферійної обробки даних (*edge*), накопичення даних, абстракції даних, прикладних сервісів та бізнес-рівень. Такий поділ дозволяє:

- формалізувати потоки даних у складних розподілених системах;
- визначити логічні межі між функціональними компонентами;
- описувати механізми масштабування в контексті великих підприємств;
- інтегрувати IoT-рішення з корпоративними інформаційними системами.

Водночас семирівнева модель Cisco орієнтована переважно на хмарні або гібридні інфраструктури, що ускладнює її пряме застосування у локальних IoT-рішеннях, де ключовими вимогами є автономність функціонування, предиктивність роботи та мінімальна залежність від зовнішніх сервісів.

1.1.2.4. Архітектура IoT-A.

IoT-A є концептуальною сервісно-орієнтованою архітектурою, у межах якої формалізовано моделі доменів, функціональних компонентів, інформаційних об'єктів, комунікаційних механізмів та безпеки. Основною перевагою IoT-A є високий рівень абстракції та формалізації, що дозволяє використовувати її як універсальний каркас для наукових досліджень і міждисциплінарних IoT-проєктів.

Водночас IoT-A не встановлює чітких вимог до реалізації edge computing, локальної обробки подій, інтеграції пристроїв різних виробників і механізмів оптимізації роботи автономних платформ. У зв'язку з цим у практичних реалізаціях архітектура IoT-A зазвичай використовується як концептуальна основа, а не як безпосередня інженерна модель.

1.1.3. Порівняльний аналіз сучасних IoT-архітектур і критерії оцінювання.

Архітектурні моделі IoT-систем, що розглядаються у сучасних наукових дослідженнях, суттєво відрізняються за рівнем деталізації, орієнтацією на хмарні або локальні середовища, а також за підходами до забезпечення безпеки та інтероперабельності. У більшості робіт архітектура подається у вигляді багаторівневих структур, які відображають логічний поділ функцій між пристроями, комунікаційною інфраструктурою, підсистемами обробки даних, сервісами та користувацькими інтерфейсами.

Окрему групу становлять архітектурні підходи, орієнтовані на локальні IoT-розгортання, що функціонують автономно або з обмеженим доступом до хмарних сервісів. Такі рішення є особливо актуальними для приватних, промислових та офісних середовищ, у яких ключовими вимогами виступають контроль над даними, зниження затримок, підвищення надійності та мінімізація зовнішніх залежностей. У межах зазначених підходів особлива увага приділяється ролі комунікаційного рівня, який забезпечує взаємодію між гетерогенними пристроями з використанням протоколів *Zigbee*, *MQTT*, *REST API* та інших механізмів локальної інтеграції [36].

У низці робіт [37–39] підкреслюється доцільність чіткого розмежування сервісного рівня, відповідального за реалізацію бізнес-логіки та автоматизаційних сценаріїв, і рівня презентації, який забезпечує взаємодію з користувачем. Такий поділ сприяє підвищенню масштабованості системи, спрощує її супровід і дозволяє незалежно розвивати інтерфейсні та сервісні компоненти [40, 41]. Загалом, запропоновані в літературі архітектурні рішення концептуально узгоджуються з відомими еталонними моделями, зокрема *Cisco IoT Reference Model* та *IoT-A*, проте демонструють різний ступінь адаптації до умов локального функціонування, вимог безпеки та автономності.

Оцінювання архітектур IoT-систем у наукових дослідженнях, як правило, здійснюється за сукупністю узагальнених критеріїв, які відображають як функціональні, так і експлуатаційні характеристики систем:

- *ефективність* - здатність архітектури забезпечувати обробку подій у режимі, близькому до реального часу, з мінімальними затримками;
- *енергоспоживання* - відповідність вимогам автономних сенсорних вузлів та оптимізація мережевої взаємодії;
- *масштабованість* - підтримка зростання кількості пристроїв, сервісів і обсягів телеметричних даних;
- *безпека* - наявність механізмів шифрування, автентифікації, контролю доступу та захисту від типових загроз IoT-середовища;
- *інтероперабельність* - здатність інтегрувати пристрої різних виробників і підтримувати відкриті протоколи;
- *керованість* - доступність засобів моніторингу, діагностики та оновлення компонентів системи;
- *надійність* - стійкість до збоїв і здатність автономно виконувати критичні

функції.

Узагальнення сучасних архітектурних підходів свідчить про еволюцію IoT-моделей від спрощених трирівневих схем до розгорнутих багаторівневих структур, орієнтованих на забезпечення безпеки, масштабованості та інтероперабельності [33, 42, 43]. Результати проведеного літературного аналізу дозволяють сформулювати ключові вимоги до архітектур локальних IoT-систем і створюють концептуальне підґрунтя для подальшого обґрунтування архітектурних рішень у наступних розділах дисертації.

1.2. Стандартизація та протоколи для IoT-систем

Розвиток Internet of Things (IoT) відбувається в умовах відсутності єдиного універсального стандарту, що зумовлює формування складної екосистеми взаємодіючих протоколів, специфікацій та архітектурних підходів і суттєво ускладнює забезпечення інтероперабельності між компонентами системи [44]. На відміну від класичних інформаційних систем, IoT характеризується високою гетерогенністю апаратних платформ, обмеженими обчислювальними ресурсами кінцевих вузлів, різноманітністю мережевих технологій та динамічністю топологій. Це ускладнює уніфікацію механізмів взаємодії та обумовлює необхідність багаторівневої стандартизації, орієнтованої на різні аспекти функціонування IoT-систем [25, 35].

Стандартизація в IoT виконує подвійну роль: з одного боку, вона забезпечує технічну сумісність між пристроями різних виробників, а з іншого — формує концептуальні принципи побудови масштабованих, безпечних і керованих систем у довгостроковій перспективі.

Інститут інженерів з електротехніки та електроніки (IEEE) є основним розробником стандартів фізичного та канального рівнів [45]. Найбільш значущими для IoT є стандарти серії IEEE 802, зокрема IEEE 802.11 (Wi-Fi) та IEEE 802.15.4, що визначає основу для низькопотужних бездротових мереж, на яких базуються протоколи Zigbee та Thread [46, 47]. Ці стандарти регламентують параметри доступу до середовища передавання, структуру кадрів, механізми енергозбереження та вимоги до апаратної реалізації, закладаючи базові технічні обмеження для побудови IoT-пристроїв.

Розробкою мережевих і транспортних протоколів для IoT займається Internet Engineering Task Force (IETF). Основний акцент робиться на адаптації класичних інтернет-протоколів до середовищ з обмеженими ресурсами, що дозволяє зберег-

ти сумісність із глобальною IP-інфраструктурою без істотного збільшення накладних витрат. Ключовим прикладом є 6LoWPAN, який забезпечує інкапсуляцію IPv6-пакетів у кадри IEEE 802.15.4 та дозволяє інтегрувати сенсорні мережі в IP-інфраструктуру [34,48]. Окрім цього, IETF розробляє рекомендації щодо маршрутизації, транспортного обміну та прикладних протоколів для constrained-вузлів.

Міжнародний союз електрозв'язку (ITU) розглядає IoT у ширшому контексті розвитку глобальних телекомунікаційних систем. Документи ITU формують термінологічні основи, узагальнені архітектурні моделі та стратегічні напрями впровадження IoT у промислових, міських і соціальних середовищах [49–51]. На відміну від IEEE та IETF, діяльність ITU має переважно концептуальний і рекомендаційний характер [52, 53].

Забезпечення інтероперабельності між гетерогенними IoT-платформами є завданням галузевих альянсів і консорціумів. IPSO Alliance зосереджується на використанні Internet Protocol як універсальної основи для IoT-взаємодії та сприяє поширенню IP-сумісних рішень у вбудованих системах [49]. Open Connectivity Foundation (OCF) та World Wide Web Consortium (W3C) розробляють підходи до уніфікації моделей даних, сервісно-орієнтованої взаємодії та інтеграції IoT із веб-технологіями, що є важливим для побудови складних прикладних сервісів.

Функціональна багатошаровість IoT-систем зумовлює необхідність класифікації протоколів за рівнями взаємодії. На канальному рівні застосовуються бездротові технології Wi-Fi, Bluetooth Low Energy (BLE), Zigbee, Thread та LoRaWAN, які відрізняються за дальністю зв'язку, пропускною здатністю, енергоспоживанням і топологією мереж [46, 47, 54, 55]. Вибір конкретної технології залежить від вимог до автономності, щільності розміщення вузлів та сценаріїв використання [56–59].

Для інтеграції малопотужних пристроїв у IP-мережі використовується протокол 6LoWPAN, який забезпечує сумісність IPv6 з обмеженими каналами передавання даних [34]. Це дозволяє застосовувати стандартні механізми адресації та маршрутизації в IoT-середовищах, зберігаючи при цьому низькі накладні витрати.

Транспортний рівень у IoT представлений протоколами TCP та UDP. TCP забезпечує надійне доставлення даних, проте супроводжується більшими накладними витратами, тоді як UDP орієнтований на мінімізацію затримок і часто використовується в поєднанні з прикладними протоколами, які самостійно реа-

лізують механізми надійності [34]. Вибір транспортного протоколу визначається компромісом між продуктивністю, енергоспоживанням і вимогами до цілісності даних, що є критичним для автономних та ресурсообмежених IoT-вузлів.

На сесійному та прикладному рівнях використовуються спеціалізовані IoT-протоколи обміну повідомленнями, серед яких MQTT, CoAP, AMQP та XMPP [35]. Вони реалізують різні моделі взаємодії та забезпечують масштабований обмін даними між великою кількістю вузлів.

Особливе місце серед них займає MQTT (Message Queuing Telemetry Transport), який базується на моделі publish/subscribe та оптимізований для передачі телеметричних даних у мережах з низькою пропускною здатністю та нестабільними з'єднаннями [34, 35]. MQTT характеризується простою структурою повідомлень, низькими накладними витратами та підтримкою різних рівнів якості обслуговування, що дозволяє адаптувати протокол до різних сценаріїв використання. Завдяки цим властивостям MQTT широко застосовується як у хмарних, так і в локальних IoT-системах, зокрема як базовий протокол обміну телеметричними даними між сенсорними вузлами та керуючими компонентами [60–62].

Не менш важливу роль відіграє протокол Zigbee, який ґрунтується на стандарті IEEE 802.15.4 та призначений для створення енергоефективних mesh-мереж сенсорних і виконавчих пристроїв [47]. Zigbee забезпечує підтримку великої кількості вузлів, механізми самовідновлення мережі та мінімальне енергоспоживання, що робить його придатним для автономних IoT-систем із батарейним живленням. Поєднання Zigbee на канальному рівні з MQTT на прикладному рівні дозволяє формувати гнучкі та керовані локальні IoT-архітектури.

Окремим напрямом сучасної стандартизації є технологія Matter - відкритий стандарт, спрямований на уніфікацію взаємодії IoT-пристроїв різних виробників [63]. Matter орієнтований на забезпечення сумісності між існуючими екосистемами та спрощення інтеграції пристроїв у межах локальних і гібридних IoT-середовищ. Його поява розглядається як важливий крок у подоланні фрагментації протоколів, яка історично ускладнювала міжплатформну інтеграцію та масштабування IoT-систем [64].

Загалом, стандартизація та багаторівнева організація протоколів є фундаментальною передумовою розвитку IoT як цілісної технологічної парадигми. Аналіз ролі організацій стандартизації та ієрархії протоколів дозволяє систематизувати підходи до побудови IoT-систем, обґрунтувати вибір комунікаційних рішень для

різних сценаріїв використання та створює теоретичну основу для подальшого аналізу архітектурних і практичних аспектів реалізації IoT-платформ.

1.3. Аналіз сучасних комунікаційних технологій IoT

Комунікаційні технології становлять фундаментальний компонент Internet of Things (IoT), оскільки саме вони визначають не лише архітектурні можливості системи, але й характер взаємодії між пристроями, допустимі затримки передавання даних, рівень енергоспоживання та потенціал масштабованості мережі [46, 47, 54, 55, 63, 64].

З огляду на багатошаровий характер IoT-систем, комунікаційні технології доцільно розглядати в контексті їхнього місця у мережевому стеку. Такий підхід дозволяє систематизувати різні протоколи та стандарти відповідно до функцій, які вони виконують у процесі передавання, маршрутизації та обміну даними між пристроями.

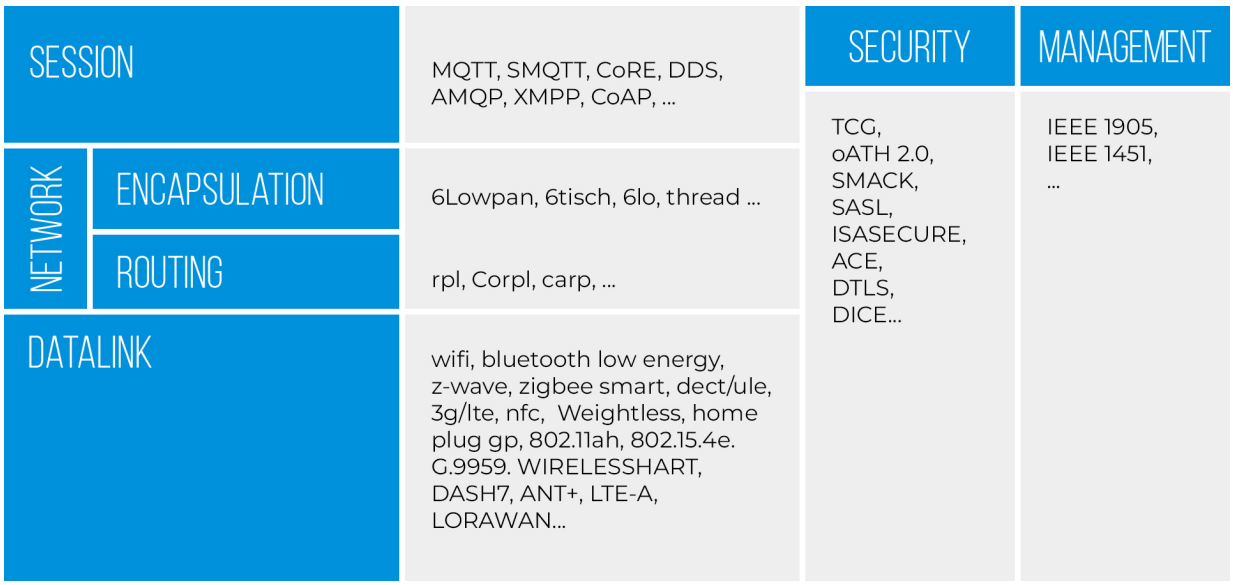

Рис. 1.2: Узагальнена структура комунікаційних протоколів IoT за рівнями

На рис. 1.2 подано узагальнений стек комунікаційних протоколів IoT, що відображає розподіл технологій за рівнями - від канального та мережевого до сесійного й прикладного [65–68]. Така класифікація підкреслює відсутність універсального комунікаційного рішення для всіх IoT-сценаріїв і демонструє, що ефективна IoT-система зазвичай формується як комбінація кількох протоколів, кожен з яких оптимізований під окремий функціональний рівень і специфічні умови експлуатації [69–71].

На канальному рівні домінують бездротові технології з різними характеристиками дальності, пропускної здатності та енергоспоживання. Саме на цьому рівні закладаються ключові обмеження щодо автономності вузлів, топології мережі та щільності розміщення пристроїв. Вище, на мережевому та транспортному рівнях, розташовуються протоколи, відповідальні за IP-сумісність, маршрутизацію та доставку пакетів, тоді як прикладний рівень визначає логіку обміну повідомленнями між компонентами системи [72].

З практичної точки зору, найбільш поширені комунікаційні технології IoT доцільно розглядати з урахуванням типу мережі та сфери застосування - локальні, персональні, глобальні або спеціалізовані мережі з низьким енергоспоживанням.

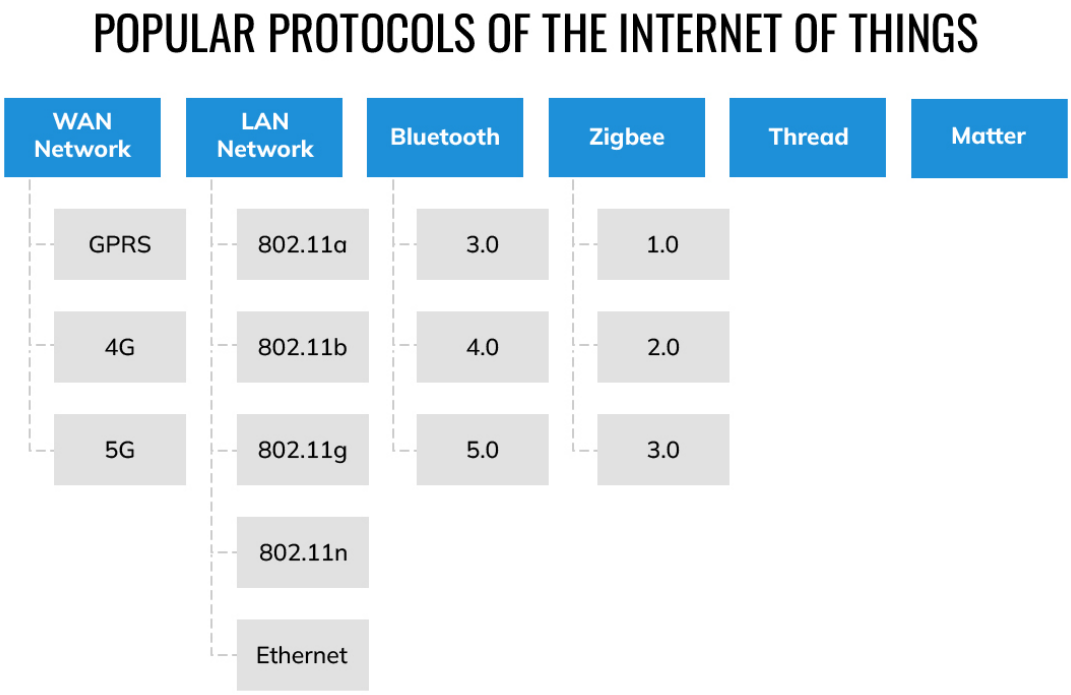

Рис. 1.3: Сучасні комунікаційні технології та протоколи Інтернету речей

Рисунок 1.3 ілюструє найбільш поширені комунікаційні технології IoT, згруповані за типами мереж та стандартами. Зокрема, Wi-Fi та Ethernet залишаються базовими рішеннями для стаціонарних пристроїв із постійним живленням, тоді як Bluetooth Low Energy, Zigbee та Thread орієнтовані на автономні сенсорні мережі з мінімальним енергоспоживанням. Окрему нішу займають LPWAN-технології та стільникові мережі, призначені для віддалених і мобільних сценаріїв.

Wi-Fi характеризується високою пропускною здатністю та низькими затрим-

ками, що робить його придатним для камер відеоспостереження, мультимедійних систем і центральних шлюзів. Водночас значне енергоспоживання обмежує застосування Wi-Fi у батарейних сенсорних вузлах. Bluetooth Low Energy (BLE), навпаки, оптимізований для коротких сеансів обміну даними та тривалої роботи в режимі сну, але має обмежену дальність і не забезпечує масштабованої mesh-топології.

Thread є IP-орієнтованою mesh-технологією, що поєднує низьке енергоспоживання з підтримкою IPv6, завдяки чому спрощується інтеграція з мережевими стеком та сучасними протоколами прикладного рівня. Однак практичне впровадження Thread наразі стримується обмеженою кількістю доступних комерційних пристроїв та відносною зрілістю екосистеми, що знижує його поширення у порівнянні з більш усталеними стандартами.

LoRaWAN належить до класу LPWAN і забезпечує передачу невеликих обсягів даних на великі відстані з мінімальними енергетичними витратами. Це робить його придатним для моніторингових і розподілених систем, проте висока затримка та низька пропускна здатність обмежують застосування у сценаріях автоматизації в реальному часі. Стільникові мережі 4G і 5G забезпечують глобальну доступність та високу швидкість передавання, але їх використання супроводжується підвищеним енергоспоживанням, а також залежністю від операторської інфраструктури, що обмежує автономність і контроль над локальною IoT-системою [68, 73, 74].

Порівняльні характеристики основних комунікаційних технологій IoT з урахуванням типу мережі, пропускної здатності та енергоефективності наведено в табл. 1.1.

Таблиця 1.1: Порівняння комунікаційних технологій IoT за типом мережі, пропускною здатністю та енергоефективністю.

Технологія	**Тип мережі**	**Пропускна здатність**	**Енерго-споживання**	**Типові сценарії використання**
Wi-Fi	WLAN / LAN	Висока	Високе	Камери, мультимедіа, стаціонарні вузли з живленням від мережі.
Bluetooth Low Energy (BLE)	WPAN	Низька-середня	Дуже низьке	Носимі пристрої, маячки; короткі транзакції та режим сну.

Таблиця 1.1: Продовження таблиці 1.1

Технологія	**Тип мережі**	**Пропускна здатність**	**Енерго-споживання**	**Типові сценарії використання**
Zigbee	LR-WPAN / Mesh	Низька	Низьке	Автономні сенсорні мережі, Smart Home; mesh-маршрутизація.
Thread	LR-WPAN / Mesh	Низька	Низьке	IP-орієнтовані IoT-системи; акцент на інтероперабельність.
LoRaWAN	LPWAN / WAN	Дуже низька	Дуже низьке	Моніторинг на великі відстані; рідкісна передача малих пакетів.
4G / 5G	Стільникова WAN	Висока-дуже висока	Високе	Мобільні пристрої, транспорт, резервні канали зв'язку.

Аналіз розглянутих комунікаційних технологій свідчить, що для локальних автономних систем домашньої та офісної автоматизації визначальними є такі характеристики, як низьке енергоспоживання кінцевих вузлів, підтримка mesh-топології та здатність до стабільного функціонування за умов обмеженого або відсутнього доступу до зовнішніх мереж. У цьому контексті протокол Zigbee вирізняється поєднанням енергоефективної передачі даних, самовідновлюваної мережевої структури та широкої номенклатури сумісних сенсорних і виконавчих пристроїв. Сукупність зазначених властивостей зумовлює широке застосування Zigbee в автономних системах домашньої та офісної автоматизації та обґрунтовує доцільність його використання як однієї з базових комунікаційних технологій при побудові локальних IoT-платформ із підвищеними вимогами до енергоефективності, надійності та незалежності від зовнішніх сервісів.

1.4. Огляд комерційних IoT-екосистем

Комерційні IoT-екосистеми сформували окремий клас рішень, орієнтованих на масовий ринок та швидке впровадження систем домашньої й офісної автоматизації [75–77]. Архітектурно такі платформи базуються на централізованих хмарних сервісах, які забезпечують керування пристроями, зберігання телеметричних даних, виконання сценаріїв автоматизації та інтеграцію з мобільними застосунками [78, 79]. Подібний підхід спрощує користувацький досвід, проте водночас накладає істотні обмеження на автономність, гнучкість і контроль над даними, що є принципово важливим у контексті дослідження локальних і приватних IoT-систем [25, 35, 80].

Однією з найбільш поширених комерційних платформ є екосистема *Xiaomi Mi Home*. Вона пропонує широкий асортимент сенсорних та виконавчих пристроїв, зокрема модулі кліматичного моніторингу, безпеки, освітлення й енергоспоживання. Архітектура платформи побудо-

вана навколо централізованої хмарної інфраструктури, через яку реалізуються автоматизації, сценарії та обробка подій. Локальна логіка керування має допоміжний характер і функціонує переважно як проксі між пристроями та хмарною інфраструктурою, що знижує автономність системи.

До основних переваг Mi Home належать:

- широкий вибір доступних за вартістю пристроїв;
- простота первинного налаштування;
- інтеграція з популярними голосовими асистентами;
- стабільна робота у типових побутових сценаріях.

Водночас платформа має суттєві недоліки:

- повна залежність від хмарної інфраструктури;
- обмежений доступ до низькорівневих API;
- регіональні обмеження інтеграцій;
- ускладнене використання в дослідницьких і експериментальних системах.

Екосистема *Apple HomeKit* орієнтована на забезпечення високого рівня безпеки та приватності користувача. Вона використовує сертифікацію пристроїв, наскрізне шифрування та жорсткі вимоги до виробників апаратних компонентів. Частина автоматизацій може виконуватися локально, однак загальна архітектура залишається тісно інтегрованою з пропрієтарною екосистемою Apple, що обмежує можливості розширення та дослідницької модифікації системи.

Перевагами HomeKit є:

- підвищений рівень криптографічного захисту;
- часткова підтримка локального виконання сценаріїв;
- тісна інтеграція з мобільними пристроями користувача.

Серед недоліків слід відзначити:

- закритість архітектури та API;
- обмежений перелік сумісних пристроїв;
- високу вартість апаратних компонентів;
- низьку придатність для експериментальних досліджень.

Платформа *Google Home* реалізує модель глибокої хмарної інтеграції з широкою підтримкою сторонніх виробників. Основна логіка обробки подій, аналітики та керування виконується у віддалених дата-центрах, що забезпечує масштабованість і зручність використання, але суттєво знижує автономність системи та унеможливлює її повноцінне функціонування за відсутності зовнішнього мережевого доступу.

До її переваг належать:

- висока сумісність із різними пристроями;
- розвинені хмарні сервіси аналітики;
- підтримка складних сценаріїв автоматизації.

Недоліками є:

- повна залежність від інтернет-з'єднання;
- відсутність повноцінної локальної логіки;
- обмежений контроль над зберіганням і обробкою даних.

Окрему категорію становить система *AJAX*, орієнтована на охоронні та сигналізаційні рішення. Її архітектура є максимально закритою та спеціалізованою, з жорстко визначеним набором пристроїв і централізованим керуванням, що обмежує застосування системи за межами охоронних сценаріїв.

Переваги AJAX:

- висока надійність у межах охоронних сценаріїв;
- оптимізація під безпекові задачі;
- стабільна робота у заданій конфігурації.

Недоліки:

- закрита пропрієтарна архітектура;
- відсутність гнучкої інтеграції;
- непридатність для дослідження загальних IoT-архітектур.

Екосистема *Tuya Smart* займає проміжне положення між жорстко закритими та умовно відкритими платформами. Вона підтримує велику кількість пристроїв різних виробників і надає частково відкриті API, проте ключові механізми обробки подій і виконання автоматизацій залишаються хмароорієнтованими, що обмежує можливість повністю локального розгортання системи [47, 54].

Її переваги:

- мультивендорна підтримка;
- гнучка інтеграція з комерційними продуктами;
- масштабованість хмарної інфраструктури.

Недоліки:

- залежність від зовнішніх сервісів;
- обмежена прозорість логіки автоматизацій;
- складність повністю локального використання.

Таблиця 1.2: Узагальнене порівняння комерційних IoT-екосистем

Платформа	Хмарна залежність	Відкритість API	Інтеграція	Придатність для локальних досліджень
Xiaomi Mi Home	Висока	Обмежена	Часткова	Низька
Apple HomeKit	Середня	Обмежена	Обмежена	Низька
Google Home	Висока	Обмежена	Висока	Низька
AJAX	Висока	Закрита	Низька	Дуже низька
Tuya Smart	Висока	Часткова	Висока	Обмежена

Таким чином, аналіз комерційних IoT-екосистем свідчить, що за умов орієнтації на масового користувача та хмарні сервіси такі рішення мають низку архітектурних обмежень, зокрема у частині автономності, відкритості та контролю над обробкою даних. Зазначені особливості

зумовлюють потребу у розгляді альтернативних підходів до побудови IoT-систем, орієнтованих на локальне розгортання, автономне виконання логіки керування та підвищений рівень приватності й контролю над даними.

1.5. Локальні серверні IoT-платформи

Локальні серверні IoT-платформи формують окремий клас програмних рішень, орієнтованих на автономне функціонування, повний контроль над процесами обробки даних та гнучку інтеграцію різнорідних пристроїв. На відміну від комерційних екосистем, такі платформи реалізують децентралізований підхід до керування, за якого ключові функції автоматизації, зберігання телеметрії та прийняття рішень виконуються безпосередньо у локальному середовищі [81, 82]. Це зумовлює їхню придатність для дослідницьких задач, експериментальних інсталяцій та систем із підвищеними вимогами до приватності, надійності й керованості [83, 84].

Локальні платформи тісно пов'язані з концепціями edge computing та low-latency automation, оскільки мінімізують залежність від зовнішніх мережевих сервісів і дозволяють обробляти події безпосередньо на сервері автоматизації [85]. Водночас їх використання передбачає підвищену відповідальність за адміністрування, інформаційну безпеку та супровід інфраструктури, що принципово відрізняє такі рішення від готових хмароорієнтованих платформ [81, 86].

Серед найбільш поширених локальних серверних платформ, що розглядаються в науковій літературі та практичних реалізаціях, виділяють Home Assistant, OpenHAB та Domoticz, які реалізують різні архітектурні підходи та орієнтовані на різні категорії користувачів і сценарії використання [87–89].

1.5.1. Home Assistant. Home Assistant є відкритою платформою автоматизації, спроєктованою як універсальний центр інтеграції IoT-пристроїв із чітким фокусом на локальне керування та збереження приватності користувача [90–92]. Архітектура системи побудована за модульним принципом і передбачає розділення між ядром платформи, інтеграційними компонентами та користувацькими автоматизаціями, що забезпечує масштабованість і розширюваність системи.

Платформа підтримує широкий спектр комунікаційних протоколів і стандартів, зокрема Zigbee, Z-Wave, Bluetooth, Thread, Wi-Fi та MQTT, що дозволяє реалізувати мультипротокольну інтеграцію пристроїв різних виробників у межах єдиного локального середовища [93]. Керування системою здійснюється через веб-інтерфейс, мобільні застосунки або зовнішні сервіси керування, що забезпечує зручність експлуатації та конфігурації [94].

Особливу увагу в Home Assistant приділено моделі безпеки. Дані за замовчуванням зберігаються локально, віддалений доступ не активується автоматично, підтримується двофакторна автентифікація користувачів та детальний контроль доступу до ресурсів і інтеграцій [95]. Такий підхід забезпечує високий рівень digital privacy і суттєво знижує ризики несанкціонованого доступу або витоку інформації.

Платформа підтримує декілька варіантів інсталяції — від спеціалізованої операційної системи до контейнеризованих і ручних розгортань, що дозволяє адаптувати її до різних апаратних платформ і сценаріїв використання [96]. Активний розвиток Home Assistant підтверджується

галузевими відзнаками та наявністю широкої міжнародної спільноти розробників і користувачів [97, 98].

Переваги Home Assistant:

- повністю локальне виконання логіки автоматизацій;
- широка підтримка протоколів і пристроїв різних виробників;
- гнучка модель інтеграцій і розширень;
- орієнтація на приватність і безпеку даних;
- активна спільнота та регулярні оновлення;
- підтримка різних апаратних архітектур.

Недоліки Home Assistant:

- відносно висока складність первинного налаштування;
- потреба у регулярному адмініструванні та оновленнях;
- зростання вимог до апаратних ресурсів при масштабуванні системи.

1.5.2. Domoticz. Domoticz є локальною серверною платформою автоматизації, орієнтованою на компактні та енергоефективні інсталяції з мінімальними вимогами до апаратних ресурсів [99]. Архітектура системи реалізована у вигляді монолітного серверного застосування з веб-інтерфейсом керування, що забезпечує централізований збір даних, базове керування пристроями та виконання автоматизацій у межах локального середовища.

Платформа підтримує інтеграцію з поширеними комунікаційними протоколами, зокрема Z-Wave, Zigbee, Wi-Fi та Bluetooth, що дозволяє застосовувати її у типових сценаріях домашньої автоматизації [100]. Основний акцент у Domoticz зроблено на простоту розгортання та швидкий запуск системи, що зумовлює її популярність серед користувачів без поглибленої технічної підготовки.

Автоматизації в Domoticz реалізуються у вигляді подієво-орієнтованих сценаріїв, які формуються на основі часових тригерів, станів сенсорів або логічних умов. Водночас механізми побудови складних багаторівневих сценаріїв залишаються обмеженими порівняно з більш модульними платформами. Розширення функціональності здійснюється через плагіни та зовнішні скрипти, що вимагає додаткових знань і не завжди має уніфікований або стандартизований характер [101].

З точки зору дослідницьких застосувань Domoticz доцільно розглядати як lightweight-рішення для прототипування або демонстраційних інсталяцій. Обмежена масштабованість, менш гнучка архітектура та застарілий користувацький інтерфейс знижують його придатність для побудови складних експериментальних систем і проведення багатопротокольних досліджень.

Переваги Domoticz:

- низькі апаратні вимоги;
- простота розгортання;
- базова підтримка популярних протоколів;
- придатність для простих локальних інсталяцій.

Недоліки Domoticz:

- обмежена масштабованість;
- застарілий користувацький інтерфейс;

- менш активна спільнота розробників;
- обмежені можливості для реалізації складних сценаріїв автоматизації.

1.5.3. OpenHAB. OpenHAB (Open Home Automation Bus) є універсальною платформою автоматизації, реалізованою на основі Java та орієнтованою на інтеграцію гетерогенних пристроїв і сервісів у межах єдиного програмного середовища [102]. Архітектура платформи базується на сервісно-орієнтованому підході, у межах якого ключову роль відіграють так звані bindings — модулі, що забезпечують взаємодію з конкретними протоколами, пристроями або зовнішніми сервісами [103].

Використання Java-платформи забезпечує операційну незалежність OpenHAB та можливість розгортання на різних апаратних і програмних платформах. Це робить систему привабливою для корпоративних або напівпромислових сценаріїв, у яких важливою є стандартизація середовища виконання та довгострокова підтримка [104]. Водночас така архітектурна модель ускладнює початкове налаштування системи та зумовлює підвищені вимоги до обчислювальних ресурсів.

Автоматизації в OpenHAB реалізуються через правила, сценарії та логічні конструкції, що дозволяють описувати складні взаємозв'язки між подіями, станами та часовими умовами. Платформа надає широкі можливості для конфігурації, однак значна частина налаштувань потребує ручного редагування конфігураційних файлів, що знижує зручність використання для користувачів без відповідної технічної підготовки [105].

З точки зору дослідницьких застосувань OpenHAB є потужним, але складним інструментом. Його доцільно застосовувати у випадках, коли необхідна глибока кастомізація та інтеграція з нестандартними системами, однак складність архітектури та менш активна екосистема обмежують ефективність платформи для швидкого прототипування й експериментальних досліджень.

Переваги OpenHAB:

- кросплатформеність;
- розширюваність через bindings;
- підтримка різних сценаріїв автоматизації.

Недоліки OpenHAB:

- висока складність конфігурації;
- залежність від Java-середовища виконання;
- повільніше освоєння платформи для нових користувачів.

1.5.4. Порівняльний аналіз локальних серверних IoT-платформ. Для узагальнення відмінностей між розглянутими локальними серверними IoT-платформами доцільно виконати їх порівняльний аналіз за ключовими характеристиками, що мають принципове значення в контексті дослідницьких та експериментальних систем. До таких характеристик належать архітектурна гнучкість, підтримка мультипротокольної взаємодії та придатність платформи до побудови й відтворення експериментальних IoT-інсталяцій.

Архітектурна гнучкість визначає здатність платформи адаптуватися до різних сценаріїв автоматизації, розширювати функціональність і модифікувати внутрішню логіку системи

без суттєвих обмежень. Мультипротокольність характеризує можливість одночасної інтеграції пристроїв і сервісів, що використовують різні комунікаційні протоколи. Придатність для досліджень відображає ступінь відкритості системи, рівень контролю над даними, повторюваність експериментів і зручність налагодження.

Зведені результати порівняння локальних серверних IoT-платформ Home Assistant, OpenHAB та Domoticz наведено в табл. 1.3.

Таблиця 1.3: Порівняльна характеристика локальних серверних IoT-платформ.

Платформа	**Архітектурна гнучкість**	**Мульти-протокольність**	**Локальна автономність**	**Поріг складності**	**Типові сценарії використання**
Home Assistant	Висока	Висока	Висока	Середній	Експериментальні IoT-системи, складні сценарії автоматизації, дослідницькі інсталяції.
OpenHAB	Висока	Середня	Висока	Високий	Кросплатформені системи з глибокою кастомізацією та нестандартними компонентами.
Domoticz	Низька	Середня	Висока	Низький	Базові сценарії автоматизації, ресурсоефективні системи.

Таким чином, порівняльний аналіз локальних серверних IoT-платформ показує, що за умови спільної орієнтації на автономне функціонування та локальну обробку даних розглянуті системи демонструють суттєві відмінності за рівнем архітектурної гнучкості, підтримкою мультипротокольної взаємодії та порогом складності експлуатації. Зазначені характеристики безпосередньо впливають на можливість використання платформ у дослідницьких та експериментальних IoT-системах.

Платформа Domoticz характеризується низьким порогом входу та помірними вимогами до апаратних ресурсів, що робить її доцільною для реалізації базових сценаріїв автоматизації. Водночас обмежена архітектурна гнучкість і менш розвинені механізми розширення ускладнюють застосування цієї платформи для складних експериментальних конфігурацій і відтворюваних досліджень.

OpenHAB, у свою чергу, забезпечує високий рівень конфігураційної гнучкості та підтримку різноманітних сценаріїв інтеграції завдяки системі bindings і кросплатформеній архітектурі. Разом із тим підвищений поріг складності налаштування та експлуатації знижує ефективність використання платформи в задачах, що потребують швидкого прототипування та оперативної модифікації експериментальних сценаріїв.

На цьому тлі Home Assistant вирізняється збалансованим поєднанням модульної архітектури, високого рівня мультипротокольної інтеграції та орієнтації на локальне виконання логіки керування. Поєднання відносно помірного порогу складності з широкими можливостями розширення та активною екосистемою інтеграцій створює сприятливі умови для побудови, відтворення та масштабування експериментальних IoT-інсталяцій. Саме ці властивості визначають доцільність використання Home Assistant як базової платформи для подальших досліджень локальних автономних IoT-систем.

1.6. Безпека IoT-систем: сучасні загрози та підходи до захисту

Безпека Internet of Things (IoT) є однією з ключових проблем сучасних кіберфізичних систем, що зумовлено поєднанням кількох чинників: обмежених обчислювальних ресурсів пристроїв, різноманіття комунікаційних протоколів, розподіленої структури системи та постійної взаємодії з зовнішніми мережами [106, 107]. На відміну від традиційних інформаційних систем, IoT-інфраструктури складаються з великої кількості автономних пристроїв, які часто не здатні реалізувати повноцінні механізми захисту, що істотно підвищує загальну вразливість системи до атак [25, 35, 108, 109].

Узагальнений підхід до аналізу загроз для IoT-систем представлено у переліку OWASP IoT Top-10, який охоплює найбільш поширені проблеми безпеки, характерні для сучасних IoT-рішень, зокрема:

- слабку або відсутню автентифікацію пристроїв і користувачів;
- передавання даних без належного шифрування;
- наявність небезпечних або надлишкових мережевих сервісів;
- уразливості прошивок і механізмів їх оновлення;
- недостатній контроль доступу до API та керуючих інтерфейсів.

Зазначені загрози можуть стосуватися як окремих IoT-пристроїв, так і системи загалом, особливо у випадку використання хмароорієнтованих архітектур.

Значна частина атак у IoT-системах пов'язана з використанням бездротових комунікаційних технологій. Для мереж Wi-Fi типовими є атаки типу *man-in-the-middle*, підміна точок доступу, перехоплення незашифрованого трафіку та компрометація облікових даних користувачів [110]. У випадку протоколів Zigbee та Bluetooth Low Energy найбільш поширеними є атаки, що виникають під час сполучення пристроїв, перехоплення криптографічних ключів, повторне відтворення повідомлень, а також порушення цілісності mesh-мереж [47, 54].

На прикладному рівні окрему увагу привертають загрози, пов'язані з протоколами обміну повідомленнями, зокрема MQTT [111, 112]. Використання моделі publish/subscribe та централізованих брокерів створює низку потенційних уразливостей, серед яких:

- відсутність обов'язкової автентифікації клієнтів;
- неконтрольований доступ до окремих топіків;
- передавання повідомлень у відкритому вигляді;
- можливість перехоплення або підміни даних за відсутності транспортного шифрування.

У хмарних сценаріях зазначені загрози посилюються, оскільки брокери та сервери керування розташовуються поза межами локальної інфраструктури користувача [34].

Окрему групу ризиків становлять хмароорієнтовані Smart Home-рішення, у яких логіка автоматизацій, зберігання даних та автентифікація користувачів реалізуються на стороні зовнішніх сервісів [113]. Такий підхід призводить до втрати автономності, залежності від стабільності мережевого з'єднання та зменшення рівня контролю над персональними даними. Компрометація хмарної платформи або порушення каналу зв'язку може спричинити втрату керування системою та витік конфіденційної інформації [35, 49].

Для локальних автономних IoT-систем підходи до забезпечення безпеки ґрунтуються на мінімізації зовнішніх залежностей та перенесенні критичних функцій усередину локальної інфраструктури. Одним із базових механізмів є сегментація мережі, яка передбачає ізоляцію IoT-пристроїв у окремих підмережах або VLAN з обмеженим доступом до критичних ресурсів. Такий підхід дозволяє локалізувати потенційні інциденти безпеки та зменшити наслідки компрометації окремих вузлів.

Важливу роль відіграє використання криптографічного захисту транспортного рівня, зокрема TLS/SSL для протоколів MQTT, HTTP та REST-інтерфейсів. Шифрування каналів зв'язку забезпечує конфіденційність і цілісність даних навіть у разі перехоплення трафіку [34]. Доповнюють ці механізми засоби автентифікації та контролю доступу, зокрема рольові моделі, токенна автентифікація та двофакторна перевірка користувачів.

Суттєвою перевагою локальних IoT-платформ є можливість обробки даних і виконання логіки автоматизацій без залучення зовнішніх сервісів. Це знижує ризики витоку інформації, підвищує стійкість системи до мережевих збоїв та забезпечує повний контроль над усіма етапами взаємодії між компонентами системи. Поєднання локального виконання, відкритих API та контрольованих протоколів створює основу для побудови більш

захищених і прозорих IoT-архітектур [46, 63].

Отже, аналіз загроз та підходів до захисту IoT-систем показує, що рівень безпеки таких рішень значною мірою визначається архітектурою системи та розміщенням її критичних компонентів. Хмароорієнтовані моделі підвищують зручність використання, проте водночас створюють додаткові ризики, пов'язані з передаванням і централізованою обробкою даних. Натомість локальні IoT-системи дозволяють обмежити площину атак за рахунок ізоляції мережі, локального виконання автоматизацій та застосування контрольованих механізмів доступу і шифрування.

1.7. Інтеграція IoT з машинним навчанням та аналітикою

Зі зростанням масштабів Internet of Things (IoT) суттєво збільшується обсяг даних, що генеруються сенсорними вузлами в реальному часі. Параметри мікроклімату, енергоспоживання, події безпеки, поведінкові сигнали користувачів і службова телеметрія формують складні багатовимірні набори даних, які характеризуються високою частотою надходження, нерівномірністю та наявністю шумів [114]. У таких умовах класичні порогові або правила-орієнтовані підходи до аналізу стають малоефективними, що зумовлює зростаючий інтерес до методів машинного навчання та аналітики даних у складі IoT-систем. [115, 116]

Одним із ключових напрямів інтеграції машинного навчання в IoT є прогнозування поведінки користувачів і характерних сценаріїв використання середовища. Аналіз історичних даних дає змогу виявляти повторювані шаблони присутності, активності та взаємодії з автоматизованими системами. На основі таких моделей можливе формування адаптивних сценаріїв керування, які враховують не лише поточні показники сенсорів, а й імовірні майбутні стани середовища [117]. Застосування методів кластеризації та бікластеризації дозволяє групувати часові інтервали, набори параметрів або типові поведінкові профілі, що підвищує узгодженість автоматизацій і зменшує кількість некоректних або надлишкових керуючих дій [118].

Важливим практичним завданням є виявлення аномалій у роботі IoT-систем. У розподілених сенсорних мережах відхилення можуть бути зумовлені як фізичними причинами (відмова датчика, деградація батареї, збої зв'язку), так і несанкціонованими діями або кібератаками. Методи машинного навчання дозволяють формувати модель нормальної поведінки системи на основі статистичних і кореляційних залежностей між параметрами [119]. Відхилення від такої моделі можуть автоматично інтерпретуватися як потенційні аномалії без необхідності ручного задання жорстких порогових значень, що є особливо важливим для динамічних середовищ. [120]

Окремий клас задач пов'язаний з оптимізацією енергоспоживання. Дані, отримані від лічильників, сенсорів навантаження та систем керування, дають змогу аналізувати реальні режими використання енергії, виявляти неефективні або надмірні витрати та прогнозувати пікові навантаження. Застосування методів регресійного аналізу та кластеризації створює передумови для адаптивного керування енергетичними ресурсами з урахуванням як поведінки користувачів, так і зовнішніх умов [121]. У результаті стає можливим досягнення балансу між енергоефективністю та комфортом експлуатації приміщень. [122]

Значну увагу в сучасних дослідженнях приділено задачам контролю якості середовища, зокрема моніторингу температури, вологості та концентрації CO_2. На відміну від простого реагування на перевищення допустимих значень, машинне навчання дозволяє аналізувати взаємозв'язки між параметрами середовища, кількістю присутніх осіб, вентиляційними режимами та зовнішніми факторами [123]. Це відкриває можливість переходу від реактивного до проактивного керування мікрокліматом, за якого система завчасно коригує параметри для запобігання погіршенню умов. [124]

Суттєвим аспектом інтеграції аналітики в IoT є вибір місця виконання обчислень. Традиційні хмарні підходи забезпечують високу обчислювальну потужність, однак супроводжуються затримками, залежністю від мережевого з'єднання та підвищеними ризиками витоку даних. У локальних автономних IoT-системах усе більшого поширення набувають підходи edge computing, за яких первинна аналітика та ML-алгоритми виконуються безпосередньо на локальному сервері або шлюзі [119]. Це дозволяє зменшити затримки, знизити навантаження на мережу та зберегти контроль над даними в межах приватної інфраструктури. [125]

Таким чином, поєднання IoT з методами машинного навчання та аналітики розглядається як ключовий

чинник розвитку інтелектуальних систем автоматизації. Використання ML-алгоритмів для прогнозування, виявлення аномалій, оптимізації енергоспоживання та контролю якості середовища суттєво розширює функціональні можливості IoT-платформ. У поєднанні з локальною архітектурою та edge-орієнтованими підходами це створює основу для побудови адаптивних, автономних і керованих IoT-систем, що є важливим напрямом подальших досліджень.

1.8. Невирішені частини загальної проблеми та постановка завдань дослідження

Проведений аналіз сучасних наукових досліджень, а також практичних реалізацій IoT-систем свідчить про те, що попри значний прогрес у розвитку апаратних платформ, комунікаційних протоколів і програмних засобів автоматизації, низка ключових аспектів побудови ефективних, безпечних і інтелектуально орієнтованих IoT-систем залишається недостатньо опрацьованою. Особливо це стосується локальних автономних IoT-рішень, призначених для дослідницьких, експериментальних і приватних сценаріїв застосування.

По-перше, у більшості існуючих робіт домінує хмароорієнтований підхід до побудови IoT-систем, у межах якого функції обробки даних, аналітики та керування реалізуються на стороні зовнішніх сервісів. Незважаючи на переваги такого підходу з точки зору масштабованості та простоти розгортання, він супроводжується суттєвими обмеженнями, зокрема залежністю від мережевої інфраструктури, підвищеними затримками обробки даних, а також зниженим рівнем контролю над процесами їх зберігання та використання. У наявних дослідженнях недостатньо уваги приділено архітектурним підходам, що забезпечують повноцінне локальне виконання аналітичних і керуючих функцій.

По-друге, локальні серверні IoT-платформи здебільшого розглядаються як інструменти автоматизації, тоді як їх потенціал як експериментального середовища для збору, накопичення та аналізу даних використовується обмежено. Відсутній формалізований підхід до проєктування локальних IoT-систем, який враховує вимоги до відтворюваності експериментів, масштабованості, модульності та можливості інтеграції аналітичних компонентів у єдину систему.

По-третє, інтеграція методів машинного навчання в IoT-системи у більшості випадків має фрагментарний характер і орієнтована на вирішення окремих прикладних задач, таких як прогнозування або виявлення аномалій. При цьому відсутній цілісний підхід до побудови інтегрованого аналітичного контуру, який забезпечує узгоджене застосування методів безнаглядного та наглядного навчання для аналізу, прогнозування та підтримки прийняття рішень безпосередньо у межах IoT-інфраструктури.

По-четверте, питання інформаційної безпеки IoT-систем часто розглядається ізольовано від архітектурних і аналітичних аспектів. Водночас забезпечення безпеки в умовах локальної обробки даних, інтеграції різнорідних пристроїв і використання інтелектуальних алгоритмів потребує комплексного підходу, що поєднує механізми сегментації мережі, контролю доступу, захищеного зберігання даних і безпечного виконання аналітичних процедур.

У зв'язку з викладеним, загальна наукова проблема дослідження полягає у відсутності формалізованого підходу до проєктування локальних автономних IoT-систем, який:

- визначає принципи побудови багаторівневої архітектури системи;
- забезпечує інтеграцію аналітичних методів у замкнений контур обробки даних і керування;
- гарантує узгодженість між архітектурними, аналітичними та безпековими компонентами системи.

Наукова новизна запропонованого підходу полягає у поєднанні архітектурного, аналітичного та безпекового рівнів у межах єдиної локальної IoT-платформи, орієнтованої на автономне функціонування та підтримку інтелектуального аналізу даних.

Для розв'язання зазначеної наукової проблеми у дисертаційній роботі поставлено такі основні завдання:

- виконати системний аналіз сучасних IoT-технологій, архітектурних підходів та програмних платформ з метою виявлення обмежень хмароорієнтованих рішень і обґрунтування доцільності застосування локальних автономних IoT-систем;

- дослідити та обґрунтувати вибір комунікаційних протоколів і стандартів IoT для локальних систем моніторингу з урахуванням вимог надійності, масштабованості, енергоефективності та сумісності пристроїв;
- розробити узагальнену багаторівневу архітектуру локальної IoT-системи, що забезпечує автономне функціонування, модульність, відмовостійкість, локальну обробку даних та підвищений рівень інформаційної безпеки;
- реалізувати програмно-апаратний прототип локальної IoT-системи з інтеграцією сенсорних і виконавчих пристроїв, сервісів збору та зберігання даних, а також механізмів автоматизації та керування;
- дослідити можливості локальної обробки та аналізу сенсорних даних у межах IoT-платформи на основі інтегрованого аналітичного контуру, що включає методи кластеризації, прогнозування та виявлення аномалій;
- оцінити функціональні, експлуатаційні та безпекові характеристики розробленої системи та експериментально підтвердити її ефективність порівняно з хмарними рішеннями за критеріями автономності, затримки обробки даних, рівня контролю над даними та масштабованості.

Розв'язання сформульованих завдань у подальших розділах дисертації дозволить сформувати цілісну модель локальної автономної IoT-системи з інтегрованими засобами аналітики та машинного навчання, а також експериментально підтвердити ефективність запропонованого підходу в реальних умовах функціонування.

1.9. Висновки по розділу 1

1. Здійснено системний аналіз сучасного стану розвитку Internet of Things (IoT) як класу кіберфізичних систем із урахуванням еволюції архітектурних моделей, комунікаційних протоколів, програмних платформ і напрямів прикладного використання. Показано, що IoT-системи дедалі частіше виходять за межі простих сценаріїв автоматизації та формують основу для інтелектуальних систем керування складними об'єктами й середовищами.
2. Проаналізовано основні архітектурні підходи до побудови IoT-систем, зокрема класичні багаторівневі моделі, та встановлено, що вони забезпечують загальну концептуальну рамку, однак не повною мірою враховують специфіку локальних автономних інсталяцій. Визначено, що для таких систем критичними є мінімальні затримки, локальна обробка подій, відтворюваність експериментів і незалежність від зовнішньої хмарної інфраструктури.
3. Досліджено сучасні комунікаційні протоколи та технології обміну даними в IoT-середовищах і встановлено, що їх універсальність ускладнює узгоджене використання в системах із підвищеними вимогами до енергоефективності, безпеки та детермінованої поведінки. Це обумовлює необхідність архітектурно виваженого вибору протокольних рішень у локальних IoT-системах.
4. Виконано порівняльний аналіз комерційних IoT-екосистем і локальних серверних платформ, у результаті якого показано, що хмароорієнтовані рішення, попри функціональну зрілість, обмежують можливості наукового та експериментального використання через архітектурну закритість, хмарну залежність і обмежений контроль над даними. Натомість локальні серверні IoT-платформи демонструють потенціал для побудови автономних, гнучких і контрольованих систем, проте потребують подальшого архітектурного узагальнення та формалізації.
5. Проаналізовано актуальні загрози безпеки IoT-систем і встановлено, що фрагментарні механізми захисту не забезпечують належного рівня безпеки в умовах розподілених бездротових мереж і активного обміну телеметрією. Обґрунтовано доцільність переходу до комплексних архітектурних підходів, які поєднують мережеву сегментацію, криптографічний захист, контроль доступу та локалізацію критичних функцій.
6. Розглянуто напрями інтеграції методів аналітики та машинного навчання в IoT-системи й показано, що існуючі рішення здебільшого реалізують такі методи фрагментарно та без урахування архітектурних обмежень IoT-середовища. Визначено необхідність побудови цілісного аналітичного контуру з

підтримкою edge-орієнтованих підходів і локального виконання ML-алгоритмів.

7. Узагальнення результатів огляду дозволило виявити науково обґрунтовану дослідницьку нішу, пов'язану з розробкою локальних автономних IoT-систем, що поєднують архітектурну гнучкість, підвищений рівень безпеки та інтеграцію методів аналітики й машинного навчання. Виявлені обмеження та протиріччя між існуючими підходами сформували основу для постановки задач дослідження та визначили напрям подальших розділів дисертації, присвячених архітектурному проєктуванню, формалізації, реалізації та експериментальній валідації локальної IoT-системи моніторингу й керування.

РОЗДІЛ 2
МОДЕЛЬ ТА АРХІТЕКТУРА ЛОКАЛЬНОЇ IOT СИСТЕМИ АВТОМАТИЗАЦІЇ ПРИМІЩЕНЬ

Аналіз сучасних систем автоматизації, проведений у попередньому розділі, показав, що більшість рішень орієнтовані на хмарну інфраструктуру. Хоча хмароцентричні моделі мають очевидні переваги у швидкості розгортання й доступності інтеграцій, вони супроводжуються істотними обмеженнями — зокрема, ризиками витоку даних, обмеженим контролем над інфраструктурою та залежністю від сторонніх платформ. В умовах зростання вимог до приватності, стійкості та безперервності сервісів виникає потреба у створенні автономних IoT-рішень, здатних функціонувати незалежно від зовнішніх чинників і забезпечувати інформаційний суверенітет системи.

У цьому контексті актуальним є завдання розробки гнучкої, безпечної та економічно ефективної архітектури, співставної за функціональністю з централізованими моделями. Ключовими вимогами до такої системи є локальне зберігання й обробка даних, повна автономність роботи, підтримка широкого спектра пристроїв різних стандартів і виробників, а також можливість масштабування без жорсткої прив'язки до конкретних технологічних стеків.

Серед доступних сьогодні відкритих платформ *Home Assistant* займає провідні позиції завдяки комбінації технічної зрілості, широкої екосистеми інтеграцій та активної спільноти розробників [126]. На відміну від платформ на кшталт *Domoticz* чи *openHAB*, *Home Assistant* пропонує вищу масштабованість рішень, розширені можливості кастомізації через YAML-конфігурацію, регулярні оновлення без втрати сумісності та підтримку сучасних стандартів автоматизації, таких як Matter і Thread. Крім того, можливість повністю локального розгортання, використання відкритого коду та відсутність обов'язкової прив'язки до хмарних сервісів створюють переваги для розгортання критично важливих систем, де визначальними є безпека, надійність і повний контроль користувача.

Запропонована в цьому розділі архітектура локальної IoT-системи базується на концепції Home Assistant і орієнтована на автономну обробку даних, централізоване управління пристроями та реалізацію сценаріїв автоматизації без залучення зовнішніх сервісів. Усі обчислювальні, аналітичні та зберігальні процеси організовані в межах приватної мережевої інфраструктури, що забезпечує високий рівень надійності, мінімізацію залежності від зовнішніх каналів зв'язку та конфіденційність обробки даних.

Основні результати, представлені у цьому розділі, опубліковані у роботах автора [127–132]

2.1. Архітектура та математична формалізація системи локальної автоматизації на основі Home Assistant

Серед доступних платформ з відкритим кодом у цій роботі використано *Home Assistant* як основну основу для запропонованої системи. *Home Assistant* - це широко застосовувана платформа автоматизації на базі Python, яка підтримує локальне керування, модульну інтеграцію та безпечне розгортання без залучення хмарних сервісів. Її розширювана архітектура та розвинена екосистема інтеграцій роблять платформу придатною для побудови децентралізованих, орієнтованих на конфіденційність IoT-середовищ, що є критично важливим для навчальних та дослідницьких приміщень.

Запропонована архітектура локальної системи автоматизації (рис. 2.1) проєктувалася за принципами модульності, масштабованості та функціональної декомпозиції. Це дозволяє безперешкодно адаптувати систему до змін кількості пристроїв, складності сценаріїв автоматизації та специфічних вимог експлуатації, зберігаючи стабільність, відмовостійкість і гнучкість інтеграції.

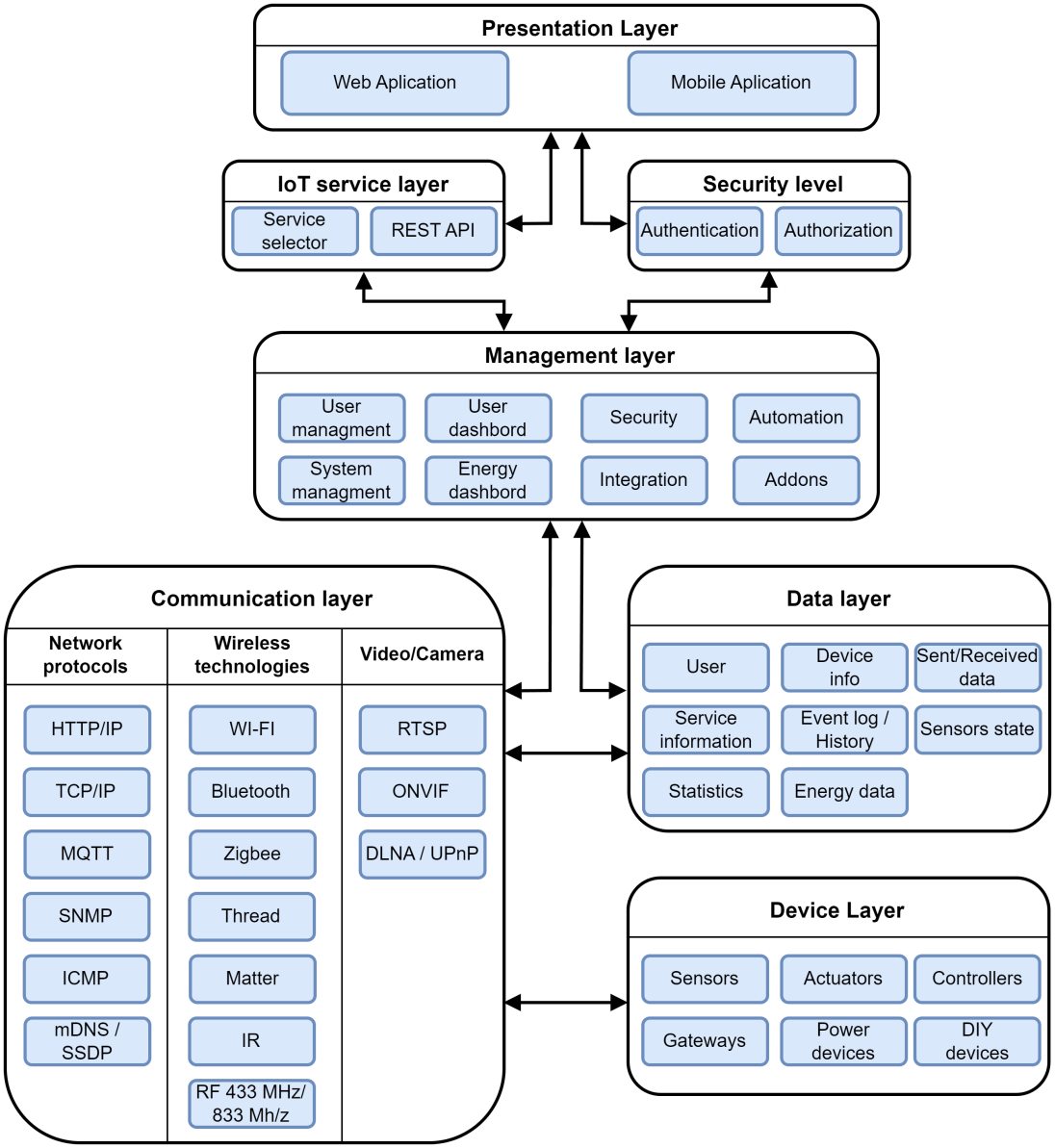

Рис. 2.1: Загальна архітектура системи локальної автоматизації на основі Home Assistant

Функціональна організація системи реалізована у форматі багаторівневої архітектури, що забезпечує чіткий розподіл компонентів за функціональними зонами відповідно до міжнародних стандартів проєктування інформаційно-керуючих систем. Архітектура включає сім взаємопов'язаних рівнів:

- **Пристрій** (Device) - фізичні сенсори, актуатори, контролери;
- **Зв'язок** (Communication) - протоколи передачі даних (MQTT, Zigbee, HTTPS);
- **Дані** (Data) - обробка, агрегація та збереження інформації;
- **Керування** (Management) - сценарії автоматизації, логіка прийняття рішень;
- **Безпека** (Security) - сегментація, шифрування, автентифікація;
- **IoT-сервіси** (IoT Service) - інтеграція з додатковими сервісами й API;
- **Презентація** (Presentation) - візуалізація та інтерфейси користувача.

Кожен рівень функціонує як незалежний, сумісний модуль зі стандартизованими інтерфейсами та чіткими обов'язками, що забезпечує узгодженість, спрощує супровід та полегшує масштабування. Така багаторівнева структура відповідає найкращим практикам сучасного проєктування систем управління, підтримуючи локалізовану обробку даних, мінімальну залежність від хмарної інфраструктури та високу сумісність із різними

класами пристроїв і протоколів.

2.1.1. Математична формалізація процесу взаємодії компонентів локальної IoT-системи.

Для опису логіки функціонування локальної IoT-системи на основі архітектури *Home Assistant* введемо багаторівневу модель, що визначається через множини та відображення. Модель формалізує взаємодію фізичних пристроїв, подій, сценаріїв автоматизації, керуючих дій та користувацького управління.

- $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ — множина пристроїв (сенсори та актуатори), кожен з яких має унікальний ідентифікатор.
- $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ — множина подій, що генеруються пристроями $d_i \in \mathcal{D}$.
- $\mathcal{G} = \{g_1, g_2, \ldots, g_k\}$ — множина шлюзів (gateways), які забезпечують трансляцію протоколів та маршрутизацію повідомлень.
- $\mathcal{S} = \{s_1, s_2, \ldots, s_p\}$ — множина сценаріїв автоматизації, реалізованих у вигляді тригерних правил.
- $\mathcal{A} = \{a_1, a_2, \ldots, a_q\}$ — множина керуючих дій, що виконуються внаслідок активації сценаріїв.
- $\mathcal{U}$ — множина користувачів системи, які взаємодіють через інтерфейс представлення.

Кожна подія e_i асоціюється з пристроєм d_j за допомогою відображення:

$$\phi : \mathcal{E} \to \mathcal{D}, \quad \phi(e_i) = d_j \tag{2.1.1}$$

Сценарій $s \in \mathcal{S}$ описується у вигляді трійки:

$$s = (\text{trigger}, \text{condition}, \text{action}), \tag{2.1.2}$$

де: *trigger* — ініціююча подія $e \in \mathcal{E}$; *condition* — булевий вираз над станами пристроїв; *action* — дія $a \in \mathcal{A}$.

Відображення між подіями та сценаріями визначається так:

$$\tau : \mathcal{E} \to \mathcal{S}, \quad \tau(e_i) = s_j \tag{2.1.3}$$

Кожна дія $a \in \mathcal{A}$ може виконуватись на одному або кількох пристроях:

$$\psi : \mathcal{A} \to 2^{\mathcal{D}}, \quad \psi(a_i) = \{d_j, d_k\} \tag{2.1.4}$$

Передача інформації про події та стани пристроїв здійснюється через шлюзи:

$$\gamma : \mathcal{D} \to \mathcal{G}, \quad \gamma(d_i) = g_l \tag{2.1.5}$$

Користувацький доступ до сценаріїв визначається відображенням:

$$I : \mathcal{U} \times \mathcal{S} \to \{0, 1\}, \tag{2.1.6}$$

де $I(u, s) = 1$, якщо користувач u має дозвіл на перегляд або керування сценарієм s.

Таким чином, наведена модель задає формальне відображення між пристроями, подіями, сценаріями та користувачами, забезпечуючи математичну основу для аналізу функціонування локальної IoT-системи та її подальшої оптимізації.

2.1.2. Оцінка ефективності локальної IoT-системи.

Оцінка ефективності є ключовим етапом аналізу архітектури локальної IoT-системи, оскільки дозволяє кількісно визначити ступінь її придатності до експлуатації в реальних умовах. У межах запропонованої системи використано багатокритеріальний підхід, що враховує основні характеристики функціонування локальної IoT-платформи: автономність, затримки обробки даних, керованість, контроль даних та безпеку.

Метрики ефективності:

- *Автономність системи (A).* Характеризує здатність системи функціонувати без зовнішніх залежностей та визначається як відношення часу безперервної роботи T_{up} до загального часу експлуатації T_{tot}:

$$A = \frac{T_{up}}{T_{tot}} \cdot 100\%. \tag{2.1.7}$$

Високі значення A свідчать про стабільність та незалежність системи.

- *Середня затримка обробки (L).* Визначає швидкодію системи та характеризує середній час між надходженням даних і їх обробкою або виконанням сценарію:

$$L = \frac{1}{N}\sum_{i=1}^{N}(t_{proc,i} - t_{in,i}), \tag{2.1.8}$$

де $t_{in,i}$ — момент надходження даних, $t_{proc,i}$ — момент завершення їх обробки.

- *Керованість системи (C).* Відображає здатність системи коректно виконувати керуючі дії та сценарії автоматизації і визначається як частка успішно виконаних операцій:

$$C = \frac{N_{exec}}{N_{req}} \cdot 100\%, \tag{2.1.9}$$

де N_{exec} — кількість успішно виконаних команд, N_{req} — загальна кількість запитів.

- *Індекс контролю даних (D).* Характеризує рівень локального контролю над даними, їх обробкою та зберіганням. Може визначатися як зважена оцінка частки даних, що обробляються локально:

$$D = \frac{V_{local}}{V_{total}}, \tag{2.1.10}$$

де V_{local} — обсяг даних, що обробляються локально, V_{total} — загальний обсяг даних у системі.

- *Індекс безпеки (S).* Визначається на основі вагової моделі, що враховує ступінь реалізації основних механізмів захисту:

$$S = \sum_{j=1}^{m}\alpha_j \cdot s_j, \qquad \sum_{j=1}^{m}\alpha_j = 1, \tag{2.1.11}$$

де s_j — нормалізовані показники (автентифікація, шифрування, контроль доступу), α_j — вагові коефіцієнти.

Слід зазначити, що окремі метрики не дають повної картини ефективності, тому вводиться узагальнений показник ефективності $\mathcal{H}$, який визначається як нормалізована зважена сума:

$$\mathcal{H} = \sum_{i=1}^{k} w_i \cdot M_i, \qquad \sum_{i=1}^{k} w_i = 1, \tag{2.1.12}$$

де $M_i \in \{A, L, C, D, S\}$ — нормалізовані значення метрик, w_i — вагові коефіцієнти.

Вибір вагових коефіцієнтів w_i залежить від прикладного сценарію:

- для систем безпеки — пріоритет мають A, C та S;
- для систем моніторингу — важливими є L та D;
- для автономних систем — ключовим є показник A.

Таким чином, запропонована модель оцінювання дозволяє формалізувати основні характеристики локальної IoT-системи та здійснювати їх кількісне порівняння у межах єдиного узагальненого показника.

2.1.3. Функціональні рівні архітектури локальної IoT-системи.

2.1.3.1. Рівень пристроїв (Device Layer).

Рівень пристроїв (Device Layer) є базовою складовою архітектури локальної IoT-системи [133,134], що забезпечує фізичну взаємодію із середовищем [135,136]. Його основні функції полягають у *сенсингу* (perception) навколишніх параметрів за допомогою сенсорів та *актууванні* (actuation) керуючих дій через актуатори [137,138]. Події, згенеровані на цьому рівні, передаються на вищі рівні архітектури для подальшого аналізу й прийняття рішень [139].

До складу цього рівня входять сенсори температури, вологості, освітленості, CO_2, руху, а також актуатори — інтелектуальні реле, димери, термостати, модулі керування HVAC. Підключення реалізовано через відкриті протоколи бездротового зв'язку: Wi-Fi, Zigbee, Z-Wave, Bluetooth Low Energy (BLE) та Thread. Такий мультипротокольний підхід формує гетерогенне IoT-середовище, дозволяє балансувати між енергоефективністю, пропускною здатністю, дальністю та надійністю комунікацій.

Ключовою особливістю є використання шлюзів відкритих стандартів (наприклад, Zigbee2MQTT, SkyConnect), інтегрованих безпосередньо з ядром Home Assistant. Це усуває залежність від закритих хмарних хабів і забезпечує прозору інтеграцію пристроїв у локально кероване середовище. Система підтримує як односпрямовані потоки даних (наприклад, сенсорні), так і двоспрямовану взаємодію (керування актуаторами), використовуючи абстракцію через *entities* Home Assistant.

Вибір протоколу залежить від вимог застосування: Zigbee та Thread забезпечують енергоощадність і підтримку mesh-топології; BLE орієнтований на малопотужні сенсори з обмеженим радіусом; Wi-Fi застосовується для пристроїв із високою потребою у пропускній здатності; Z-Wave — для стабільного зв'язку на менш зашумлених частотах.

Новизна підходу полягає в **адаптивній багатошлюзовій конфігурації**, яка реалізує принцип *plug-and-play* без залежності від хмарних сервісів. Це створює основу для масштабованої, безпечної та стійкої до відмов інфраструктури, яка формує ключові події для сценаріїв автоматизації (виявлення присутності, зміни мікроклімату, стани обладнання) і передає їх на вищі рівні системи для подальшої обробки.

2.1.3.2. Рівень комунікації (Communication Layer).

Рівень комунікації (Communication Layer) забезпечує транспортування даних між фізичними пристроями та логічними компонентами системи, підтримуючи узгоджену взаємодію між рівнем пристроїв і рівнями обробки, керування та аналізу. Його ефективність визначає загальну швидкодію, надійність і масштабованість архітектури. У межах запропонованої локальної IoT-системи він відповідає за інтеграцію різнорідних середовищ передачі даних, консолідуючи стандарти бездротового та дротового зв'язку в єдину транспортну інфраструктуру.

Передбачено підтримку мультипротокольного обміну, що забезпечує гнучкість у конфігурації мережі та адаптацію до умов експлуатації без прив'язки до конкретного фізичного носія або типу пристрою. Комунікаційний рівень абстрагує транспортне середовище, досягаючи балансу між енергоефективністю, пропускною здатністю, дальністю зв'язку та стійкістю до збоїв. Ключову роль відіграють шлюзи (gateways), що виконують трансляцію між протоколами нижнього рівня та IP-мережею серверної інфраструктури, створюючи єдину модель обміну даними між усіма компонентами.

Обмін повідомленнями реалізовано через легковаговий протокол публікації–підписки MQTT (Mosquitto), який гарантує мінімальні накладні витрати й підтримує рівні якості обслуговування (QoS). Це критично важливо для своєчасного поширення оновлень станів і тригерів подій у середовищах з обмеженою пропускною здатністю. У такій конфігурації сенсорні вузли та актуатори (ESPHome, Zigbee2MQTT) виступають як видавці MQTT, надсилаючи телеметрію локальному брокеру. *Home Assistant* одночасно функціонує як підписник (отримання оновлень і запуск автоматизацій) та видавець (керування пристроями, трансляція змін станів). На відміну від хмароцентричних платформ, комунікація залишається повністю локалізованою, що усуває залежність від зовнішніх сервісів.

Для інтерактивної взаємодії в реальному часі застосовуються WebSocket-з'єднання та RESTful API, що забезпечують двонапрямлену комунікацію на рівні застосунків. Автоматичне виявлення пристроїв і конфігураційні служби (DHCP, mDNS, Zeroconf) дають змогу швидко підключати нові вузли без ручного налаштування. Локальна топологія гарантує працездатність навіть у разі ізоляції від глобальної мережі. Таким чином, Communication Layer формує самостійно розгорнуту, відмовостійку магістраль обміну повідомленнями, що забезпечує безперервну роботу автоматизацій і масштабованість системи.

2.1.3.3. Рівень даних (Data Layer).

Рівень даних (Data Layer) виконує центральну функцію в управлінні інформаційними ресурсами IoT-системи, забезпечуючи зберігання, структурування, попередню обробку та надання доступу до даних. У межах архітектури на базі *Home Assistant* цей рівень інтегрує модулі коротко- й довгострокового зберігання, підсистеми реєстрації подій та інтерфейси взаємодії зі сценаріями автоматизації й аналітичними сервісами. Дані з пристроїв надходять у вигляді телеметрії або змін станів (state changes) та обробляються ядром Home Assistant за подієво-орієнтованою моделлю.

Кожен пристрій у системі представлений як сутність (*entity*) з унікальним ідентифікатором, постійними атрибутами та часовими мітками, що формує уніфіковану модель даних незалежно від типу або походження

пристрою. Це дозволяє будувати цілісну історію станів, виконувати аналітику та забезпечувати зворотні зв'язки для сценаріїв автоматизації. На відміну від багатьох IoT-систем, які покладаються на зовнішні сховища даних, Home Assistant надає вбудований механізм зберігання на базі SQLite, оптимізований для малих і середніх інсталяцій. За потреби можливе масштабування до продуктивніших рішень, зокрема PostgreSQL, InfluxDB або TimescaleDB, що підтримують часові ряди та великі обсяги даних. Така модульність забезпечує динамічне перенаштування серверної частини без втрати сумісності з інструментами візуалізації (наприклад, Grafana) та зовнішніми аналітичними платформами.

Функціональність Data Layer виходить далеко за межі пасивного зберігання. Він реалізує попередню обробку, фільтрацію й агрегацію даних, підтримує тригери у реальному часі та цикли зворотного зв'язку з низькою затримкою. Завдяки використанню шаблонів (Jinja2 templates) у Home Assistant можна створювати похідні сутності, обчислювати складні залежності між параметрами та запускати автоматизації безпосередньо на периферії, без звернення до хмарних обчислювальних сервісів.

Додаткові засоби *Recorder*, *History* та *Logbook* забезпечують інтерактивний перегляд подій, створення ретроспективних звітів і налаштування глибини зберігання даних. Механізми include/exclude-фільтрації та параметри частоти оновлення дозволяють оптимізувати навантаження на систему. У підсумку Data Layer виступає активним елементом як аналітичної, так і операційної підтримки, формуючи основу для адаптивної поведінки компонентів, підтримки сценаріїв автоматизації й масштабованості всієї архітектури.

2.1.3.4. Рівень керування (Management Layer).

Рівень керування (Management Layer) відповідає за реалізацію логіки автоматизації, координацію взаємодії між компонентами системи та адаптацію до змін у фізичному середовищі. В архітектурі *Home Assistant* цей рівень виконує функції ядра прийняття рішень: інтерпретує події, формує керувальні команди та забезпечує їх виконання відповідно до визначених сценаріїв. На відміну від комерційних хмарних платформ, де логіка залежить від зовнішніх оркестраторів і мережевих затримок, запропонована реалізація підтримує повністю локальне прийняття рішень, що гарантує низьку затримку реакцій і автономність навіть в умовах ізоляції від Інтернету.

Функціональність Management Layer базується на автоматизаціях, скриптах, шаблонах (templates) та інтеграціях, які визначають програмну поведінку системи через конфігураційно-керовану модель. Основу сценаріїв складає концепція «тригер–умова–дія» (trigger–condition–action), що формалізує реакцію на події різної складності: від зміни стану окремого сенсора до багаторівневих логічних конструкцій. Завдяки YAML-конфігураціям і механізму Jinja2-шаблонів кожну автоматизацію можна інтерпретувати як реактивну систему зі скінченними станами, що забезпечує відтворюваність поведінки й детальний контроль над сценаріями.

Управління сценаріями здійснюється як через текстові YAML-конфігурації для детального налаштування, так і через інтерактивний редактор, що дозволяє створювати автоматизації без програмування. Базові функції рівня включають централізовану оркестрацію, локальне виконання скриптів та синхронізацію між кластерами пристроїв. На відміну від підходів на кшталт FIWARE, які покладаються на центральний брокер контексту, у нашій архітектурі реалізовано децентралізовану логіку на периферії, що підвищує стійкість і зменшує мережеве навантаження.

У великомасштабних або багатодоменних інсталяціях особливої ваги набуває проблема динамічного виявлення та семантичної сумісності між API й сервісами. Останні дослідження пропонують використання каталогів як гармонізаційних шарів. Зокрема, фреймворк Goldie [**?**] реалізує підхід до оркестрації пристроїв на основі глобального каталогу, який може бути інтегрований у запропоновану архітектуру як міжрівневий сервіс, покращуючи адаптивність сценаріїв і динамічну оркестрацію автоматизацій.

Архітектурною перевагою є відокремлення рівня керування від рівня презентації: усі процеси прийняття рішень та виконання сценаріїв відбуваються автономно, навіть у разі недоступності інтерфейсу або мережевих ресурсів. Таким чином, Management Layer виступає логічним центром архітектури локальної IoT-системи, забезпечуючи узгоджену взаємодію між підсистемами, адаптивність до змін середовища та високий рівень автоматизації для динамічної поведінки компонентів.

2.1.3.5. Рівень безпеки (Security Layer).

Рівень безпеки (Security Layer) виконує критичну функцію забезпечення цілісності, конфіденційності та доступності компонентів і даних у межах архітектури локальної IoT-системи. У системах на базі *Home Assistant* безпека інтегрована у всі архітектурні рівні й розглядається як невід'ємна частина проєктування, а не як окремий модуль, що додається на фінальних етапах розробки.

Ключовим принципом є мінімізація векторів атаки за рахунок ізоляції від глобальної мережі, локалізації автентифікації та виконання всіх обчислювальних і зберігальних процесів у межах внутрішнього середовища. На відміну від архітектур, що покладаються на віддалені API чи хмарні брокери, запропонована система за замовчуванням не відкриває зовнішні інтеграції без явного дозволу користувача. Такий підхід підсилює цифровий суверенітет і знижує ризик витоку даних.

Технічна реалізація включає наскрізне TLS-шифрування для веб-інтерфейсів і API, використання самопідписаних сертифікатів або автоматичне керування через *Let's Encrypt* [140]. Home Assistant підтримує багаторівневу модель автентифікації з локальною реєстрацією користувачів, розмежуванням прав доступу та інтеграцією зовнішніх систем авторизації (OAuth2, LDAP). Усі інтеграції працюють за принципом «найменших привілеїв», що мінімізує ризики ескалації доступу в разі компрометації окремого компонента.

На рівні мережевої інфраструктури застосовується фізичне й логічне сегментування (VLAN), фаєрволи й VPN-доступ. Це дозволяє ізолювати IoT-пристрої від критично важливих сегментів мережі та зменшує площу атаки. Додатково реалізовано контроль доступу на основі списків дозволених і заборонених IP-адрес та ведення журналу аудиту (audit trail), що забезпечує відстежуваність і спрощує реагування на інциденти.

Таким чином, Security Layer охоплює всі рівні архітектури — від пристроїв до сценаріїв автоматизації — й забезпечує відповідність сучасним вимогам інформаційної безпеки, підвищену стійкість до зовнішніх і внутрішніх загроз та можливість розгортання системи в ізольованих середовищах із жорсткими політиками доступу.

2.1.3.6. Рівень сервісів IoT (IoT Service Layer).

Рівень сервісів IoT (IoT Service Layer) виконує роль інтеграційного прошарку, що координує обмін даними та керувальними командами між прикладною логікою автоматизації й функціональними компонентами системи. У межах архітектури *Home Assistant* він забезпечує уніфікований доступ до сервісних функцій пристроїв, обробку API-запитів і інтеграцію з внутрішніми та зовнішніми модулями незалежно від фізичних протоколів зв'язку чи топології мережі.

Ключовим інструментом цього рівня є відкриті прикладні інтерфейси, зокрема RESTful API та WebSocket, які реалізують двонаправлену асинхронну взаємодію в реальному часі. REST API дозволяє виконувати запити на читання та зміну станів пристроїв, викликати сценарії автоматизації й реєструвати події, тоді як WebSocket-інтерфейс мінімізує затримку та мережеві витрати, забезпечуючи інтерактивний обмін даними з мобільними застосунками, локальними контролерами й зовнішніми сервісами. Як уніфікований формат корисного навантаження використовується JSON, що забезпечує сумісність із системами візуалізації й оркестрації (наприклад, Node-RED, Grafana).

На відміну від монолітних архітектур централізованих брокерів, запропонований сервісний рівень підтримує мікросервісну парадигму, безстанні виклики (stateless service calls) і модульне масштабування. Це дозволяє розгортати систему у вигляді розподіленого середовища, зберігаючи цілісність даних і стабільність прикладної логіки навіть за зміни апаратного забезпечення чи мережевої конфігурації.

Крім інтеграції з внутрішніми компонентами, IoT Service Layer підтримує взаємодію з клієнтськими пристроями, голосовими асистентами та системами сповіщення, забезпечуючи як ручне, так і автоматизоване ініціювання викликів на основі тригерів і умов сценаріїв. У такий спосіб цей рівень виступає стратегічним компонентом гнучкості та еволюційного розвитку архітектури, створюючи масштабовану основу для інтеграції аналітичних і зовнішніх сервісів.

2.1.3.7. Рівень представлення (Presentation Layer).

Рівень представлення (Presentation Layer) відповідає за візуалізацію даних, організацію доступу до сервісів системи та інтерактивну взаємодію користувачів з компонентами автоматизації. В архітектурі *Home Assistant* цей рівень реалізується через інтерфейс *Lovelace UI*, який надає динамічні інформаційні панелі, кастомізовані

віджети та представлення з розмежуванням доступу за ролями. Усі елементи інтерфейсу мають реактивний характер: стани сутностей оновлюються в режимі реального часу без перезавантаження сторінок.

Lovelace UI підтримує адаптивний дизайн, темізацію, багатомовність і може працювати як у локальній мережі, так і при дистанційному доступі через захищене з'єднання. Побудова панелей виконується динамічно на основі сутностей системи, що дозволяє персоналізувати інтерфейс відповідно до профілю користувача. Інтеграція з мобільними застосунками доповнюється підтримкою геолокаційних сервісів, push-сповіщень і локального кешування даних, що підвищує доступність і зручність використання в мобільних сценаріях.

Важливою архітектурною перевагою є відокремленість Presentation Layer від рівня керування: сценарії та автоматизації виконуються автономно, навіть у разі збою або недоступності інтерфейсу. На відміну від хмароцентричних порталів, які потребують постійного підключення до зовнішніх серверів, інтерфейс Home Assistant зберігає повну функціональність у локальному режимі без доступу до Інтернету.

З міркувань безпеки використовується наскрізне HTTPS/TLS-шифрування, захищене зберігання конфігурацій та постійний кеш, що підвищує стійкість системи до мережевих збоїв. Таким чином, Presentation Layer забезпечує не лише відображення даних, а й надійний канал інтерактивного управління, формуючи позитивний користувацький досвід і підтримуючи цілісність локальної IoT-архітектури.

2.2. Функціональні підсистеми локального сервера автоматизації

Як зазначалося у попередньому підрозділі, архітектура *Home Assistant* вирізняється високою гнучкістю, модульністю та масштабованістю, що забезпечує її адаптивність до індивідуальних вимог користувачів. Відкритий код, підтримка широкого спектра пристроїв і протоколів, а також активна спільнота розробників дозволяють цій платформі охоплювати як типові сценарії побутової автоматизації, так і розширені функціональні напрями — зокрема енергоменеджмент, безпеку, контроль мікроклімату та мультимедійні сервіси. Завдяки великій кількості вбудованих інтеграцій і можливості створення власних модулів на Python, Home Assistant може функціонувати як повноцінний локальний сервер автоматизації без використання зовнішніх хмарних сервісів.

Функціональна організація такої системи представлена на рисунку 2.2, де показано ключові підсистеми, що взаємодіють із центральним сервером. До них належать: підсистема безпеки, керування енергоспоживанням, контроль мікроклімату, автоматизація освітлення, а також компоненти системної підтримки. Така структура дозволяє реалізувати модульну та масштабовану систему, у якій кожна підсистема виконує власний набір завдань і може налаштовуватися або розширюватися незалежно від інших.

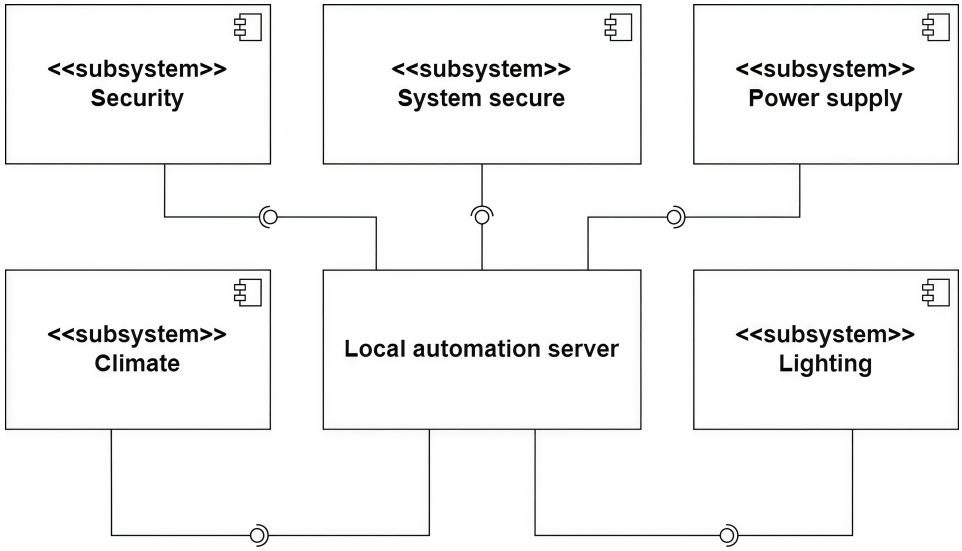

Рис. 2.2: Узагальнена модель функціональних підсистем локального сервера автоматизації на базі Home Assistant

Для повного уявлення про принципи побудови системи доцільно також розглянути логічну архітектуру програмних модулів *Home Assistant*, яка визначає функціональні блоки платформи та характер їхньої взаємодії. Як показано на рисунку 2.3, система включає чотири основні логічні блоки: моніторинг сенсорних даних (Home Sensor Monitoring), керування пристроями (Device Management), доступ до інтерфейсу (Access System) та модуль автоматизації (Automation System). Кожен із них забезпечує ключові процеси безперервної роботи сервера незалежно від масштабів інсталяції чи кількості підключених пристроїв.

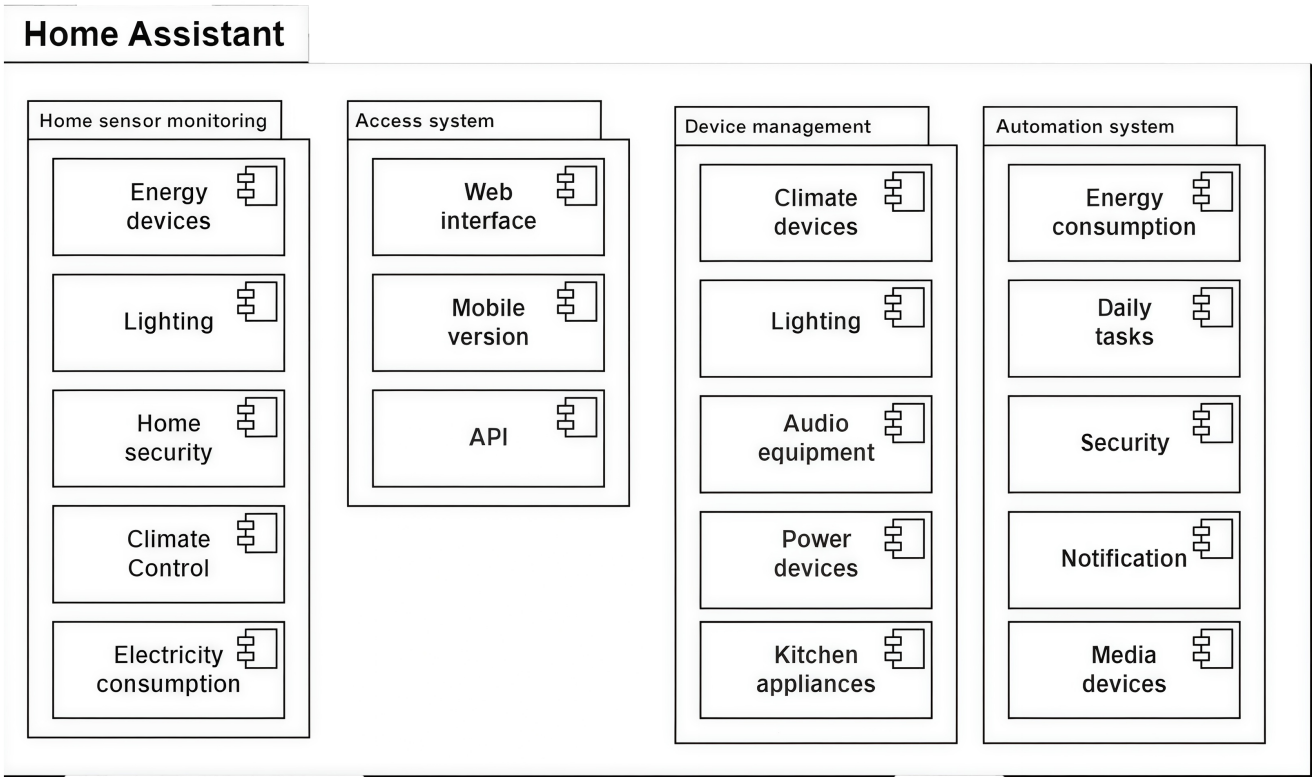

Рис. 2.3: Логічна архітектура програмних модулів системи Home Assistant

Як видно з рисунка 2.3, програмні компоненти системи тісно взаємодіють, забезпечуючи повну інтеграцію функціональності в межах локальної інфраструктури.

Функціональні підсистеми локального сервера автоматизації є прикладним відображенням архітектурних рівнів, розглянутих у попередньому підрозділі. Кожна підсистема охоплює окрему сферу життєдіяльності інтелектуального середовища — безпеку, клімат-контроль, енергоспоживання тощо — та реалізується через інтеграцію фізичних пристроїв, програмних сервісів і сценаріїв автоматизації. Їхня структура є модульною: підсистеми функціонують автономно, але водночас можуть взаємодіяти одна з одною через ядро *Home Assistant*. Нижче наведено огляд ключових функціональних підсистем, що забезпечують повноцінну роботу локальної системи автоматизації:

2.2.1. Підсистема «Безпека».

Підсистема безпеки є однією з ключових складових локального сервера автоматизації, оскільки забезпечує моніторинг стану приміщень, виявлення потенційних загроз і ініціювання дій у межах попередньо визначених сценаріїв реагування. Її архітектура побудована на принципах модульності та адаптивної інтеграції, що дозволяє поєднувати комерційні пристрої різних виробників із сенсорами власної розробки (DIY) за допомогою стандартів ESPHome, Zigbee2MQTT, Bluetooth Low Energy та MQTT.

На рисунку 2.4 наведено приклад типового компонування елементів підсистеми безпеки в середовищі *Home Assistant*. Схема демонструє взаємозв'язки між сенсорними пристроями, комунікаційними модулями, службами аналізу подій і системами сповіщення, що формують наскрізну логіку виявлення та реагування.

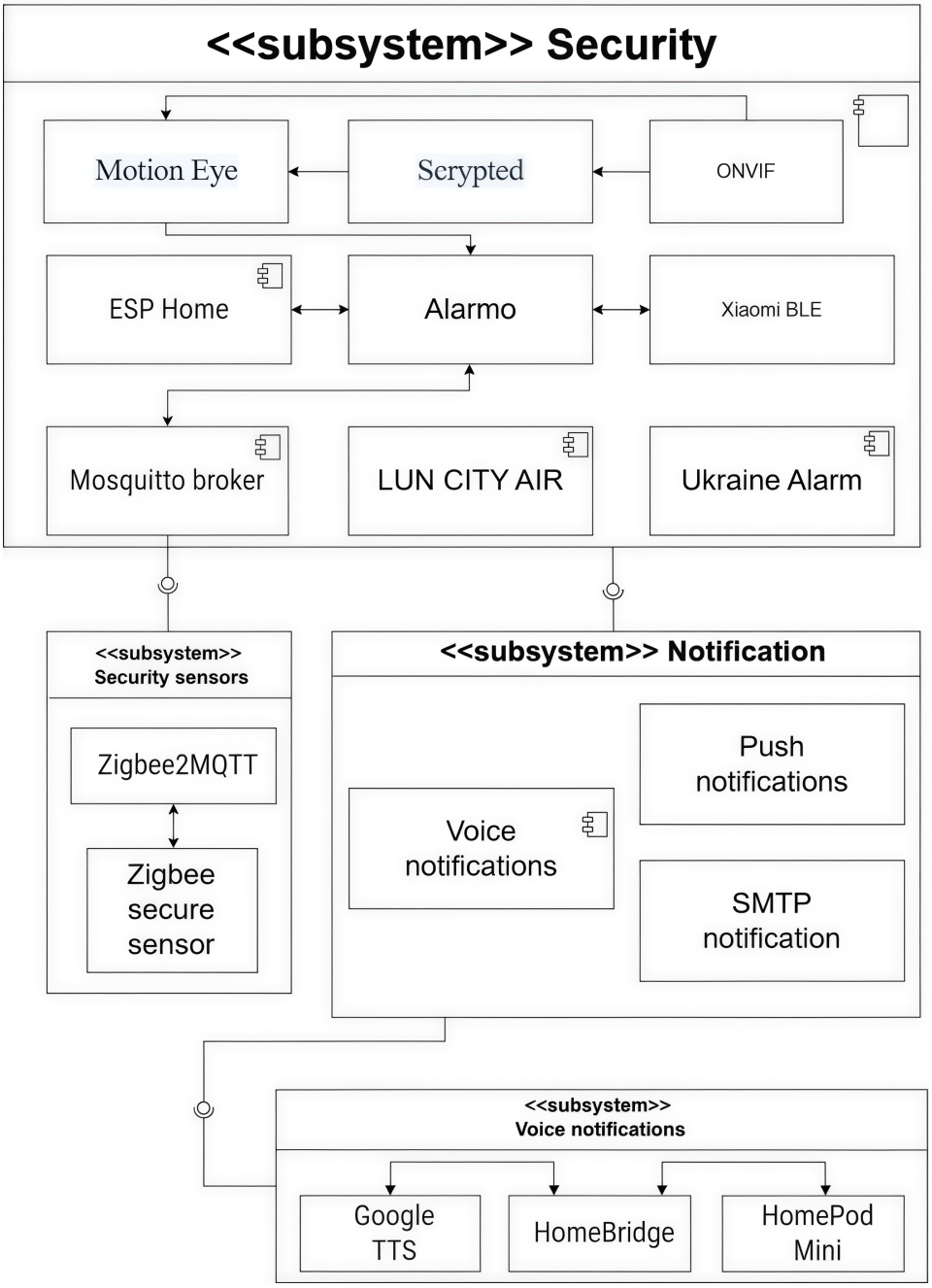

Рис. 2.4: Структура підсистеми «Безпека» в системі Home Assistant

Основні функціональні можливості підсистеми охоплюють моніторинг витоку води, відкриття дверей і вікон, вібрацій, виявлення чадного газу та інших небезпечних ситуацій. Усі події фіксуються в режимі реального часу, передаються до ядра системи та обробляються в межах подієво-орієнтованої моделі *Home Assistant*, після чого активуються відповідні сценарії автоматизації.

Інтеграція з системами відеоспостереження реалізована через підтримку стандарту ONVIF і програмного забезпечення MotionEye. Це забезпечує виявлення руху на основі відеопотоку, автоматизований запис, надсилання сповіщень та перегляд поточної ситуації через вебінтерфейс або мобільний застосунок. Така інтеграція підвищує рівень інформативності системи та скорочує час реагування на інциденти.

У межах підсистеми також реалізовано моніторинг параметрів зовнішнього середовища. Використання

відкритих джерел, таких як LUN City Air або Ukraine Alarm, дозволяє отримувати актуальну інформацію про стан повітря, рівень забруднення, хімічні чи радіаційні загрози, а також сигнали повітряної тривоги. Подібна функціональність є критичною для роботи в умовах високих ризиків.

Оповіщення користувача про події здійснюється через багатоканальну систему сповіщень, що включає push-повідомлення, електронну пошту (SMTP), голосові повідомлення (Google TTS, HomePod, HomeBridge) та локальні сигнали. Сповіщення можуть ініціюватися як у відповідь на фізичні події, так і в межах програмованих сценаріїв, що забезпечує гнучке налаштування реакцій на зміни середовища.

Огляд основних елементів, що входять до складу підсистеми безпеки, їхнє функціональне призначення, сценарії використання та способи інтеграції наведено в таблиці 2.1.

Таблиця 2.1: Компоненти та функціональні модулі підсистеми безпеки

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
ONVIF	Відкритий стандарт інтеграції IP-відеообладнання. Підтримує керування камерами, детекцію руху, подієві сповіщення та потокове відео.	Підключення камер відеоспостереження до Home Assistant, активація запису, перегляд у вебінтерфейсі.	Камери передають відео та події через ONVIF API для обробки Home Assistant.
MotionEye	Вебінтерфейс для відеоспостереження з функцією виявлення руху та налаштуванням зон спостереження.	Виявлення руху у відео, запуск запису або сценаріїв (увімкнення світла, сповіщення).	Інтегрується з Home Assistant через REST API або MQTT, передає події та зображення.
Scrypted	Платформа відеоінтеграції з підтримкою IP-камер, відеодзвінків, розпізнавання облич та сумісністю з HomeKit Secure Video.	Забезпечує підключення камер (наприклад, Reolink, UniFi) з транскодуванням, архівацією, розпізнаванням руху й інтеграцією з автоматизаціями.	Інтегрується з Home Assistant через API, MQTT або HomeKit, передаючи статуси, події та потоки відео.
ESPHome	Інструмент для створення мікропрограм для ESP8266/ESP32 з підтримкою численних сенсорів і актуаторів.	Використовується для створення сенсорів відкриття, диму, газу, протікання.	Передає дані до Home Assistant через нативний API або MQTT-брокер.

Продовження таблиці 2.1

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Xiaomi BLE	Bluetooth Low Energy сенсори компанії Xiaomi (температура, вологість, CO_2).	Моніторинг мікроклімату та безпекових параметрів; тригери для сценаріїв автоматизації.	Передають дані через BLE-шлюзи або напряму при наявності ключа шифрування.
LUN City Air	Онлайн-сервіс моніторингу якості повітря у містах України з відкритим API.	Збір екологічних даних для відображення в інтерфейсі та умов автоматизації.	API-запити передаються до Home Assistant через інтеграційний компонент.
Ukraine Alarm	Сервіс моніторингу надзвичайних ситуацій (повітряна тривога, хімічна небезпека тощо).	Реалізація сценаріїв реагування, попередження користувача, зміна поведінки системи.	Інтеграція через REST API, події використовуються у сценаріях Home Assistant.
Mosquitto Broker	MQTT-брокер з підтримкою QoS, авторизації та TLS.	Обмін повідомленнями між ESPHome, Zigbee2MQTT, сенсорами та ядром системи.	Служить центральною точкою для ретрансляції даних у Home Assistant.
Zigbee2MQTT	ПЗ для інтеграції Zigbee-пристроїв у Home Assistant через MQTT. Використовує апаратний Zigbee-адаптер і конвертує дані в стандартизовані MQTT-повідомлення.	Додавання та керування Zigbee-пристроями (датчики руху, дверей, кнопки, розетки, лампи).	Дані з адаптера передаються в MQTT-брокер, інтегруються в Home Assistant.
Alarmo	Компонент для реалізації сигналізації у Home Assistant. Підтримує режими охорони, затримки, коди доступу та сценарії тривоги.	Побудова гнучкої системи сигналізації з режимами “відсутній”, “нічний”, “додому”.	Інтегрується з датчиками, сиренами, освітленням і мобільними сповіщеннями.

Продовження таблиці 2.1

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Push Notification Service	Канал сповіщень через мобільні застосунки Home Assistant та інші сервіси.	Реакція на події: витік, рух, тривога.	Сповіщення генерується Home Assistant при виконанні умов сценаріїв.
SMTP Notification	Надсилання електронних листів через SMTP.	Повідомлення про тривоги, збої, сервісні оновлення.	Листи надсилаються через зовнішній SMTP-сервер (Gmail, Mailgun тощо).
Google TTS	Сервіс озвучування тексту через Google API.	Голосові попередження: “Рух у вітальні”, “Пожежна тривога”.	API Home Assistant надсилає текст, який відтворюється через динаміки.
HomeBridge	Інструмент інтеграції пристроїв з Home Assistant у HomeKit (Apple).	Керування пристроями через Siri, створення сцен у додатку Apple Home.	Віртуальні пристрої Home Assistant доступні у HomeKit через HomeBridge.

2.2.2. Підсистема «Клімат».

Підсистема «Клімат» є важливою функціональною складовою локального сервера автоматизації, відповідальною за моніторинг та регулювання параметрів мікроклімату в житлових, комерційних і навчальних приміщеннях. Вона забезпечує автоматизоване керування температурою, вологістю, рівнем CO_2 та іншими характеристиками повітряного середовища з метою підтримання комфортних і безпечних умов перебування.

На рисунку 2.5 наведено типовий приклад конфігурації підсистеми «Клімат» у середовищі *Home Assistant* із зазначенням основних сенсорних та виконавчих елементів, каналів інтеграції та сценарних залежностей. Схема ілюструє модульну побудову підсистеми й принципи взаємодії між її компонентами.

Архітектура підсистеми базується на модульному підході з відкритою інтеграцією, сумісною як із комерційними пристроями (Sonoff, Shelly, Aqara, Tuya), так і з сенсорами власної розробки (DIY). Це дає змогу адаптувати рішення під конкретні об’єкти з урахуванням бюджету та вимог до функціональності. Дані з сенсорів надходять до ядра системи у форматі JSON, обробляються за подієво-орієнтованою моделлю та зберігаються у базах даних часових рядів для аналізу динаміки.

Основні функції підсистеми включають керування системами опалення, вентиляції та кондиціонування (HVAC) на основі заданих сценаріїв і фактичних показників середовища. Наприклад, при зниженні температури нижче цільового значення активується підігрів, а при перевищенні — охолодження; контроль вологості реалізується шляхом автоматичного керування зволожувачами або осушувачами. Такі дії спрямовані на підтримання параметрів у межах нормативних або комфортних значень.

На відміну від традиційних автономних HVAC-систем із фіксованими алгоритмами, реалізація підсистеми «Клімат» у *Home Assistant* забезпечує вищу гнучкість, контекстну обізнаність та адаптивність. Відкрита архітектура дозволяє враховувати широкий спектр факторів — від розкладу занять і геолокації користувачів до

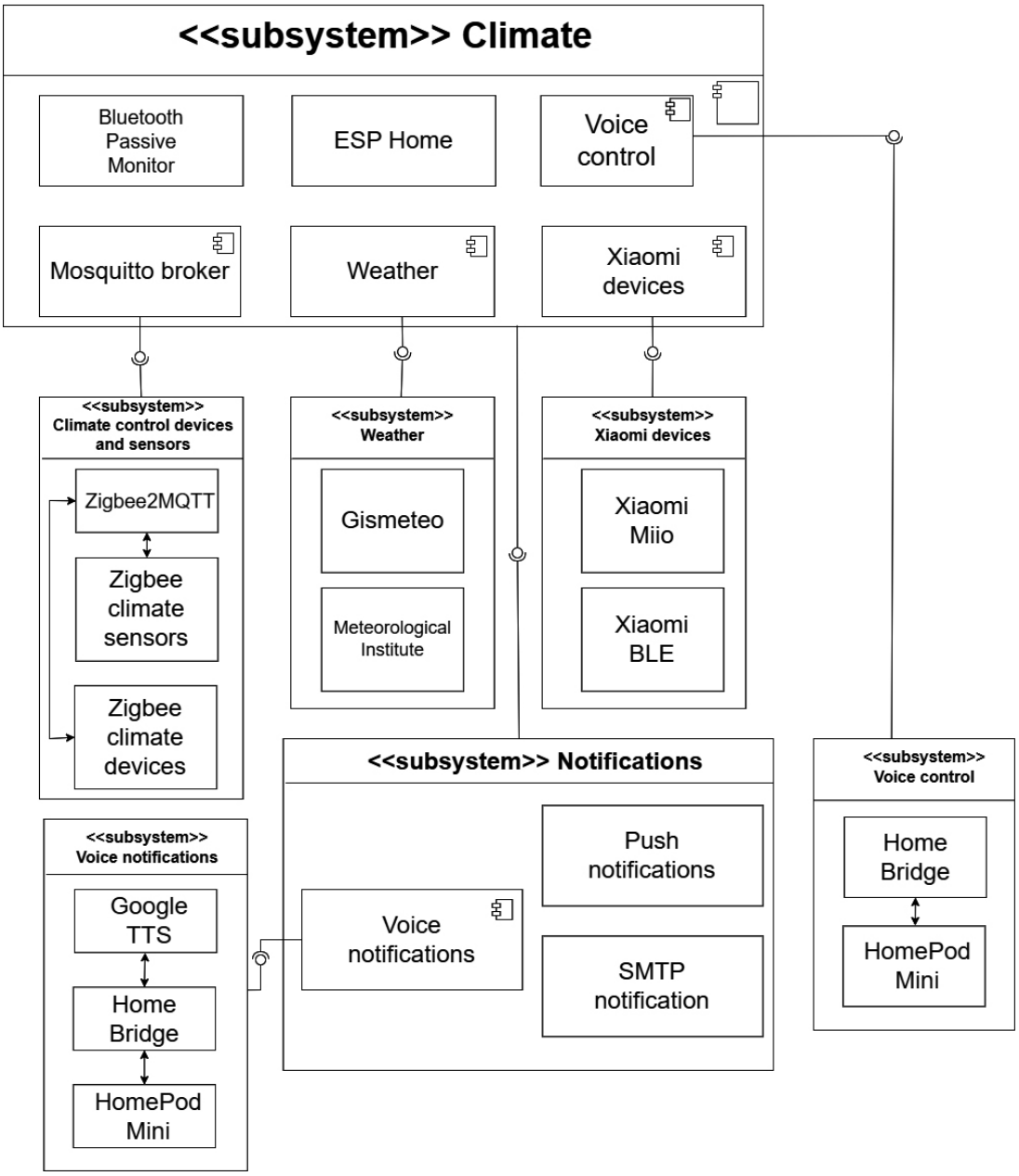

Рис. 2.5: Структура підсистеми «Клімат» в системі Home Assistant

якості зовнішнього повітря та факту присутності людей у приміщенні. Це створює передумови для персоналізованих сценаріїв енергозбереження, зниження витрат на опалення й кондиціювання та динамічного реагування на зміну умов у реальному часі.

Підсистема підтримує адаптивні механізми, що враховують зовнішні погодні умови (через погодні API), присутність користувачів (BLE-маячки, моніторинг Wi-Fi-з'єднань), а також розклад активності. Це дозволяє не лише оптимізувати енергоспоживання, а й забезпечити високий рівень персоналізації керування мікрокліматом.

Інтеграція з іншими підсистемами відкриває можливості для складних сценаріїв. Наприклад, при перевищенні допустимого рівня CO_2 в аудиторії система може активувати вентиляцію або ініціювати повідомлення про необхідність провітрювання; у разі різкого погіршення якості зовнішнього повітря — обмежити приплив, зберігаючи внутрішній комфорт.

Платформа забезпечує персоналізований зворотний зв'язок: користувачі отримують контекстно-залежну ін-

формацію через аналітичні дашборди, графіки, мобільні застосунки чи голосових асистентів. Повідомлення про відхилення параметрів або зміну режиму надсилаються з урахуванням критичності події, обраного каналу доставки та індивідуальних налаштувань — це сприяє інформованому, ненав'язливому керуванню мікрокліматом. Огляд основних елементів, що входять до складу підсистеми клімату, їхнє призначення, способи використання та взаємодії наведено в таблиці 2.2.

Таблиця 2.2: Компоненти та функціональні модулі підсистеми клімату

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Bluetooth Passive Monitor	Система пасивного моніторингу пристроїв за технологією Bluetooth Low Energy (BLE).	Виявлення присутності користувачів і зчитування даних BLE-сенсорів (температура, вологість, CO_2) для динамічної адаптації клімату; автоматичне перемикання режимів HVAC.	Дані з BLE-пристроїв надходять у Home Assistant через Bluetooth-адаптер і використовуються в кліматичних сценаріях у режимі реального часу.
Weather	Інтеграція з зовнішніми метеосервісами (наприклад, OpenWeatherMap, Gismeteo) для отримання актуальних погодних даних.	Моніторинг температури, вологості, тиску, швидкості/напряму вітру; оптимізація внутрішнього мікроклімату через адаптивне керування опаленням, охолодженням, вентиляцією або зволоженням.	Інформація автоматично інтегрується у Home Assistant і використовується у сценаріях прогнозного кліматичного керування.
Xiaomi Devices (Miio)	Компонент інтеграції з екосистемою Xiaomi через протокол Miio (двосторонній обмін).	Збір мікрокліматичних даних і керування кліматичними пристроями Xiaomi (зволожувачі, кондиціонери, термостати); реалізація гнучких сценаріїв.	Дані надходять через шлюз Xiaomi або безпосередньо з пристроїв для подальшого використання в автоматизаціях.

Продовження таблиці 2.2

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Xiaomi Devices (BLE)	Інтеграція з BLE-пристроями екосистеми Xiaomi.	Зчитування даних із BLE-сенсорів для моніторингу температури, вологості, якості повітря; активація зволожувачів/вентиляції за порогами.	Передавання даних здійснюється безпосередньо до Home Assistant через Bluetooth-адаптер для локальної обробки.
ESPHome	Інструмент з відкритим кодом для прошивок ESP8266/ESP32 з підтримкою сенсорів і виконавчих елементів.	Побудова індивідуальних пристроїв моніторингу клімату (температура, вологість, CO_2, IAQ) та керування вентиляцією/обігрівом/зволоженням.	Передавання даних до Home Assistant через нативний API або MQTT, що забезпечує швидку реакцію на зміни.
Mosquitto Broker	MQTT-брокер з підтримкою QoS, TLS-шифрування, автентифікації та ретенції повідомлень.	Передача кліматичних даних між сенсорами, ESPHome, Zigbee2MQTT та ядром Home Assistant; синхронізація сценаріїв і реакцій.	Центральний комунікаційний вузол для обміну MQTT-повідомленнями у режимі реального часу.
Zigbee2MQTT	Проміжне ПЗ для інтеграції Zigbee-пристроїв через MQTT.	Підключення Zigbee-сенсорів (температура, вологість, тиск) та виконавчих модулів (реле для обігрівачів, вентиляції, клапанів).	Дані з Zigbee-адаптера транслюються через Zigbee2MQTT до брокера Mosquitto і далі інтегруються в Home Assistant.
Push Notification Service	Канал оперативних сповіщень через мобільні застосунки Home Assistant та інші сервіси.	Миттєве інформування про відхилення параметрів (температура, вологість, тиск, CO_2).	Сповіщення формуються ядром Home Assistant на основі тригерів автоматизації й доставляються на мобільні пристрої.

Продовження таблиці 2.2

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
SMTP Notification	Канал електронних повідомлень (SMTP).	Надсилання детальних звітів або сповіщень про аномальні кліматичні події (аварії опалення, збої вентиляції тощо).	Інтегрується зі сценаріями Home Assistant, використовуючи зовнішній поштовий сервер (Gmail, Mailgun тощо).
Google TTS	Сервіс синтезу мовлення від Google для голосових повідомлень.	Голосове озвучування кліматичних подій (наприклад: «Температура у дитячій кімнаті перевищила 28 °C»).	Home Assistant передає текст через API Google TTS; звук транслюється через локальні динаміки.
HomeBridge	Платформа інтеграції *Home Assistant* з Apple HomeKit.	Керування термостатами, зволожувачами, кондиціонерами через Siri; створення кліматичних сцен в Apple Home.	Віртуальні пристрої Home Assistant експортуються до HomeKit через HomeBridge.
Voice Control	Голосове керування через Siri, Google Assistant та інші сервіси.	Зміна кліматичних параметрів (температура, режими вентиляції, вмикання обігрівача) голосовими командами.	Голосові запити інтерпретуються Home Assistant і активують відповідні дії за сценаріями.

2.2.3. Підсистема «Освітлення».

Підсистема «Освітлення» є важливою складовою локального сервера автоматизації, відповідальною за формування комфортного, функціонального та енергоефективного світлового середовища. Вона впливає не лише на ергономіку простору та суб'єктивний комфорт, а й на продуктивність і зорове навантаження. Основне призначення підсистеми полягає в автоматизованому керуванні штучним освітленням на основі аналізу зовнішніх і внутрішніх чинників: рівня природного освітлення, присутності користувачів, розкладу подій, добового циклу та індивідуальних сценаріїв.

Такий підхід дозволяє знижувати енергоспоживання й адаптувати освітлення до функціонального призначення приміщення. Наприклад, в аудиторіях або офісах світло вмикається лише під час занять чи за наявності людей, а в житлових приміщеннях інтенсивність у вечірні години зменшується для підтримання циркадних ритмів або імітації присутності за відсутності мешканців.

Функціональність підсистеми охоплює діапазон від дискретного вмикання/вимикання до динамічного керування яскравістю та корельованою колірною температурою (CCT) залежно від часу доби й типу активності (робота, навчання, відпочинок). Використання сенсорів руху, освітленості, відкриття дверей та «розумних» вимикачів забезпечує реактивну поведінку системи. Наприклад, за перевищення порога природної освітленості

штучне світло автоматично пригасає або вимикається.

Інтеграція підсистеми освітлення в *Home Assistant* відкриває широкі можливості для сценарного керування з урахуванням взаємозв'язків з іншими підсистемами. Зокрема, активація освітлення може залежати від присутності (за даними сенсорів руху чи геолокації), стану вікон (контакти) або режиму мультимедійної системи (автоматичне затемнення під час роботи проєктора).

Завдяки підтримці відкритих протоколів (Zigbee, Z-Wave, MQTT, ESPHome, Modbus) *Home Assistant* забезпечує інтеграцію різнорідних пристроїв від багатьох виробників в єдину логічну інфраструктуру. Користувачі можуть створювати сцени, наприклад «Навчання», «Перерва», «Зустріч», що активують узгоджені комбінації освітлення. Керування здійснюється автоматично або вручну через мобільні застосунки, сенсорні панелі чи голосових асистентів (Google Assistant, Amazon Alexa).

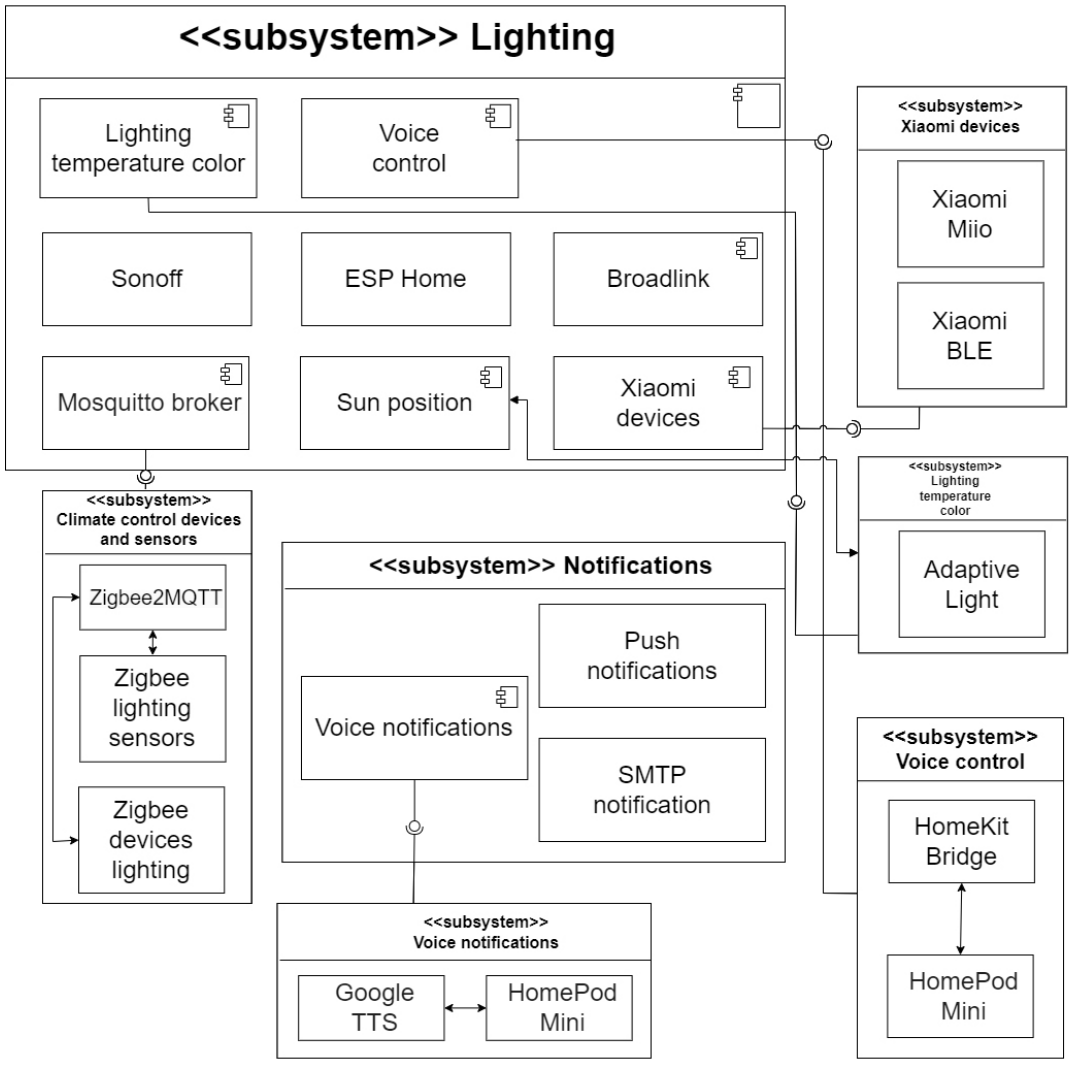

Рис. 2.6: Структура підсистеми «Освітлення» в системі Home Assistant

Підсистема підтримує адаптивне освітлення, що змінює не лише рівень яскравості, а й спектральні характеристики світла: у ранкові години переважає «холодне» біле світло для підвищення бадьорості, у вечірні — «тепле» для зменшення зорового навантаження та підтримання біоритмів. Для навчальних просторів це сприяє підвищенню концентрації та зниженню втоми.

Отже, підсистема освітлення в архітектурі *Home Assistant* є гнучким і масштабованим компонентом, що поєднує енергоощадність, ергономіку та індивідуалізовані сценарії. Вона може масштабуватися від базових кон-

фігурацій до багаторівневих систем із розгалуженою логікою автоматизації. Огляд основних елементів, їхнього призначення, сценаріїв використання та способів інтеграції наведено в таблиці 2.3.

Таблиця 2.3: Компоненти та функціональні модулі підсистеми освітлення

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Sonoff	Пристрої керування живленням (реле, вимикачі) з підтримкою Wi-Fi або RF.	Вмикання/вимикання освітлення та електроприладів за розкладом або вручну.	Інтеграція з Home Assistant через MQTT або власний API.
Sun (position)	Компонент визначення положення Сонця (схід, захід, висота).	Автоматизація освітлення залежно від часу доби та природної освітленості.	Дані використовуються в автоматизаціях Home Assistant через внутрішні сенсори «Sun».
Broadlink	IR/RF-пристрої для бездротового керування сумісною технікою, зокрема освітленням.	Керування лампами/світильниками з ІЧ/РЧ-пультами.	Інтеграція через Broadlink API, емуляція IR/RF-команд.
Xiaomi Devices (Miio)	Інтеграція з екосистемою Xiaomi через протокол Miio (двосторонній обмін).	Керування лампами, світильниками, вимикачами; сценарії: розклад, вечірнє приглушення, вимкнення за відсутності руху.	Зв'язок через шлюз Xiaomi або напряму; події використовуються в автоматизаціях.
Xiaomi Devices (BLE)	Інтеграція з BLE-пристроями екосистеми Xiaomi.	Управління BLE-світильниками; зміна яскравості/кольору залежно від часу доби чи освітленості; увімкнення за детекцією руху.	Передавання даних до Home Assistant через Bluetooth-адаптер для локальної обробки.

Продовження таблиці 2.3

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
ESPHome	Фреймворк прошивок ESP8266/ESP32 з підтримкою сенсорів і актуаторів.	Кастомні контролери освітлення (димери, сенсорні вимикачі, детектори присутності/освітленості); автономні рішення під специфіку об'єкта.	Інтеграція через MQTT або нативний API ESPHome з низькою затримкою виконання.
Adaptive Lighting	Програмний модуль адаптивного освітлення.	Автоматична зміна яскравості та CCT за добовим циклом; сцени для навчання/відпочинку.	Працює на базі даних «Sun» і системного часу; інтегрується зі «смарт»-лампами.
Mosquitto Broker	MQTT-брокер (QoS, TLS, автентифікація, ретенція).	Стабільний обмін подіями між освітлювальними пристроями, сенсорами, вимикачами та ядром Home Assistant.	Центральний вузол маршрутизації MQTT-повідомлень у реальному часі.
Zigbee2MQTT	Проміжне ПЗ для інтеграції Zigbee-пристроїв через MQTT.	Керування лампами, диммерами, реле, сенсорами присутності; авто-виявлення нових пристроїв; сценарне керування.	Zigbee-адаптер → Zigbee2MQTT → Mosquitto → Home Assistant.
Push Notification Service	Канал швидких сповіщень через мобільні застосунки або зовнішні сервіси.	Повідомлення про залишене ввімкнене світло, нічну активацію за рухом, недоступність керування.	Сповіщення генерує Home Assistant на основі тригерів автоматизації.
SMTP Notification	Канал структурованих електронних повідомлень (SMTP).	Звіти про стан освітлення, завершення «вечірніх»/«ранкових» сценаріїв, повідомлення про збої.	Листи надсилаються через зовнішній поштовий сервер, інтегрований у Home Assistant.

Продовження таблиці 2.3

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Google TTS	Сервіс синтезу мовлення для голосових повідомлень.	Озвучування подій освітлення («Нічний режим активовано», «Світло у спальні вимкнено»).	Home Assistant передає текст через API; відтворення на локальних динаміках.
HomeBridge	Міст інтеграції з Apple HomeKit.	Керування освітленням через Siri; сцени в Apple Home; геолокаційні сценарії.	Експорт віртуальних пристроїв Home Assistant у HomeKit через HomeBridge.
Voice Control	Голосове керування через Siri/Google Assistant тощо.	Вмикання/вимикання, димування, зміна кольору голосовими командами.	Запити інтерпретуються Home Assistant і активують відповідні дії за сценаріями.

2.2.4. Підсистема «Енергоживлення».

Підсистема «Енергоживлення» в архітектурі *Home Assistant* реалізується за модульним принципом з використанням відкритих протоколів зв'язку, що забезпечує сумісність як із комерційними рішеннями (EcoFlow, Sonoff, Shelly, APC), так і з компонентами власної розробки — зокрема вимірювальними модулями на основі ESP32, NUT-серверами та DIY-джерелами резервного живлення. Такий підхід дозволяє адаптувати конфігурацію системи до особливостей об'єкта з урахуванням наявної інфраструктури, бюджету та вимог до надійності живлення окремих вузлів.

Компоненти підсистеми об'єднуються в єдину систему через MQTT, Modbus, Zigbee або REST API, передаючи дані до ядра у форматі JSON для подальшої обробки, візуалізації та автоматизованого прийняття рішень. Основні функції охоплюють моніторинг електричних параметрів (напруга, струм, потужність, частота), контроль якості живлення та оцінювання поточних навантажень — як у зональному, так і в пристроєвому розрізі. Дані з цифрових лічильників і сенсорів зберігаються у базах часових рядів, що уможливлює виявлення трендів, аналіз профілів навантаження та прогнозування критичних ситуацій. У разі відхилення параметрів або аварії система може автоматично вимикати другорядні навантаження чи перемикатися на резервні джерела — акумуляторні станції, ДБЖ або генератори.

На відміну від статичних енергетичних рішень із жорстко фіксованими алгоритмами, запропонована реалізація підтримує сценарії керування з урахуванням зовнішніх чинників: тарифних зон, графіків активності користувачів, погодних умов або статусу інших підсистем. Це відкриває можливості для режимів енергозбереження (demand response): зменшення потужності у пікові години, перемикання на економніші джерела, обмеження навантаження вночі тощо.

Контекстна обізнаність системи забезпечує адаптивну поведінку в реальному часі та баланс між енергоефективністю і стабільністю роботи. Користувацький інтерфейс надає дашборди, індикатори стану, звіти та графіки. Сповіщення про аномалії — перевантаження, зникнення живлення, критичний розряд — доставляються через мобільні застосунки, електронну пошту чи месенджери з урахуванням пріоритетності подій і налаштувань користувача.

Інтеграція з іншими підсистемами дає змогу реалізовувати узгоджені реакції. Наприклад, у разі зникне-

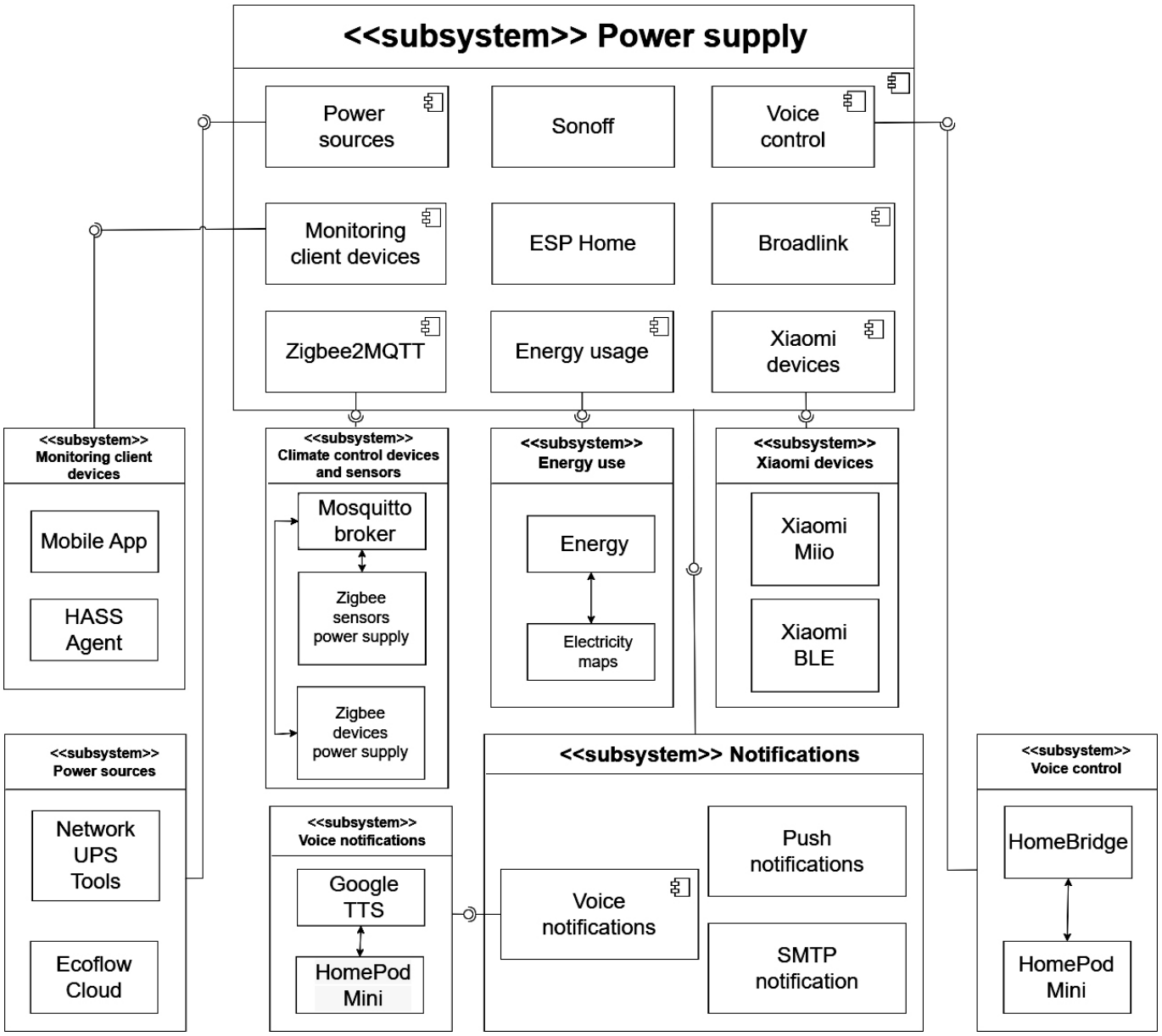

Рис. 2.7: Структура підсистеми «Енергоживлення» в системі Home Assistant

ння зовнішнього живлення система автоматично оптимізує режим клімату, обмежує освітлення та переводить охоронні компоненти в енергозберігаючий режим. Таким чином, підсистема «Енергоживлення» формує основу стабільного, керованого й безпечного функціонування всієї локальної інфраструктури. Огляд основних елементів, їхнього призначення, сценаріїв використання та способів інтеграції наведено в таблиці 2.4.

Таблиця 2.4: Компоненти та функціональні модулі підсистеми енергоживлення

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Power sources	Джерела живлення (акумулятори, сонячні панелі, мережа, гібридні рішення).	Моніторинг напруги, заряду, споживання, ефективності джерел; оцінка автономності.	Інтеграція через фірмові API або USB/Modbus; дані у *Home Assistant* для аналітики енергоспоживання.
EcoFlow Cloud*	Хмарний сервіс для моніторингу та керування портативними станціями EcoFlow.	Отримання рівня заряду, потужності, часу автономної роботи; налаштування режимів.	REST API-інтеграція з *Home Assistant* для відображення параметрів у локальних дашбордах.
Network UPS Tools (NUT)	Моніторинг ДБЖ різних виробників.	Контроль рівня батареї, напруги, частоти; сценарії на випадок зникнення електроживлення.	Інтеграція через USB/мережу; тригери аварійних сценаріїв у *Home Assistant*.
Sonoff	Розумні реле/вимикачі (Wi-Fi/RF) з енергомоніторингом.	Керування живленням, вимкнення за розкладом/умовами, контроль споживання.	Інтеграція через MQTT або фірмовий API; телеметрія у ядро *Home Assistant*.
Monitoring client devices	Моніторинг енергоспоживання клієнтських пристроїв у ЛОМ.	Дані про активність/споживання для ПК, NAS, ТВ тощо.	Збір через SNMP, MQTT, Shelly, ESPHome або HASS Agent; подальша візуалізація.
Mobile App	Мобільний застосунок *Home Assistant*.	Доступ до енергетичних дашбордів, push-сповіщень, сценаріїв живлення.	Зв'язок через API; керування та сповіщення в реальному часі.

Продовження таблиці 2.4

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
HASS Agent	Агент для Windows з двостороннім зв'язком.	Контроль енергоспоживання ПК, системні події, стани сну/живлення.	Інтеграція через MQTT або REST; запуск автоматизацій за подіями.
Energy usage	Функціональність *Home Assistant* для агрегації енергоданих.	Відстеження споживання, ефективності та побудова сценаріїв оптимізації.	Інтеграція даних Energy/Electricity Maps/«розумних» лічильників у дашбордах.
Energy	Вбудований компонент статистики енергоспоживання.	Моніторинг споживання за зонами/пристроями, виробництва (PV), вартості.	Підтримка розеток, інверторів, відновлюваних джерел; візуалізації.
Electricity Maps	Дані про мікс генерації та вуглецевий слід електроенергії.	Оптимізація часу споживання за екологічним критерієм.	API-інтеграція з *Home Assistant* для корекції сценаріїв.
Broadlink	IR/RF-керування технікою, що має ПДУ.	Вмикання/вимикання навантажень (обігрівачі, бойлери, кондиціонери).	Інтеграція через Broadlink API; емулювання IR/RF-команд.
Xiaomi Devices (Miio)	Інтеграція з екосистемою Xiaomi (розетки, вимикачі).	Моніторинг/керування, відключення за лімітом споживання чи відсутності активності.	Зв'язок через шлюз Xiaomi або напряму; події в автоматизаціях.
Xiaomi Devices (BLE)	Інтеграція з BLE-пристроями Xiaomi.	Керування BLE-розетками, енергомоніторинг; відключення за відсутності навантаження.	Передавання даних через Bluetooth-адаптер для локальної реакції.

Продовження таблиці 2.4

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
ESPHome	Фреймворк для ESP8266/ESP32 (сенсори струму, реле, енергомоніторинг).	Кастомні рішення: реле з контролем навантаження, відсічення за перевищення потужності/температури.	Обмін даними через MQTT або нативний API; низька затримка реагування.
Mosquitto Broker	MQTT-брокер (QoS, TLS, автентифікація, ретенція).	Стабільний обмін між розетками, лічильниками, сенсорами споживання та ядром.	Центральний вузол маршрутизації MQTT-повідомлень для енергомоніторингу.
Zigbee2MQTT	Проміжне ПЗ для інтеграції Zigbee-пристроїв енергоконтролю.	Підключення розеток, реле з енергомоніторингом, сенсорів струму/напруги.	Zigbee-адаптер → Zigbee2MQTT → Mosquitto → *Home Assistant*.
Push Notification Service	Канал оперативних сповіщень (мобільні застосунки, месенджери).	Алерти про перевантаження, просідання напруги, розряд акумулятора, зникнення живлення.	Сповіщення формуються ядром *Home Assistant* на основі тригерів моніторингу.
SMTP Notification	Канал структурованих електронних повідомлень (SMTP).	Періодичні звіти, повідомлення про завершення сценаріїв живлення, помилки пристроїв.	Надсилання через зовнішній поштовий сервер; асинхронна доставка відповідальним особам.
Google TTS	Сервіс синтезу мовлення для аудіоповідомлень.	Голосове інформування: «Резервне живлення активовано», «Споживання перевищує норму».	Передавання тексту через API; відтворення на локальних динаміках/смарт-колонках.
HomeBridge	Міст до Apple HomeKit.	Відстеження станів і сценарне керування живленням через Siri/Apple Home.	Експорт віртуальних пристроїв *Home Assistant* у HomeKit через HomeBridge.

Продовження таблиці 2.4

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Voice Control	Голосове керування енергоспоживанням.	Вмикання/вимикання навантажень, перевірка статусу живлення, активація режимів економії.	Команди інтерпретуються *Home Assistant* і активують відповідні сценарії.

Слід зазначити, що використання EcoFlow Cloud є опційним, не впливає на локальну автономність і за потреби дозволяється лише з явного дозволу користувача.

2.2.5. Підсистема «Системні служби».

Підсистема «Системні служби» є невід'ємною частиною інфраструктури моніторингу та керування, відповідальною за забезпечення безперервності, надійності та інформаційної безпеки функціонування локального сервера автоматизації. Її завдання виходять за межі збору технічних параметрів: підсистема формує цілісне адаптивне середовище, здатне своєчасно виявляти відхилення, прогнозувати критичні стани та ініціювати профілактичні дії.

Ключовими функціями є безперервне спостереження за ресурсами хост-системи та середовищем виконання: завантаженням CPU, використанням пам'яті й дисків, станом мережевих інтерфейсів, доступністю інтеграцій. Постійний контроль дозволяє оперативно фіксувати перевантаження, аномалії виконання та деградацію продуктивності, а також формувати сигнали для автоматизованих реакцій (масштабування, перезапуск служб, зміна політик логування).

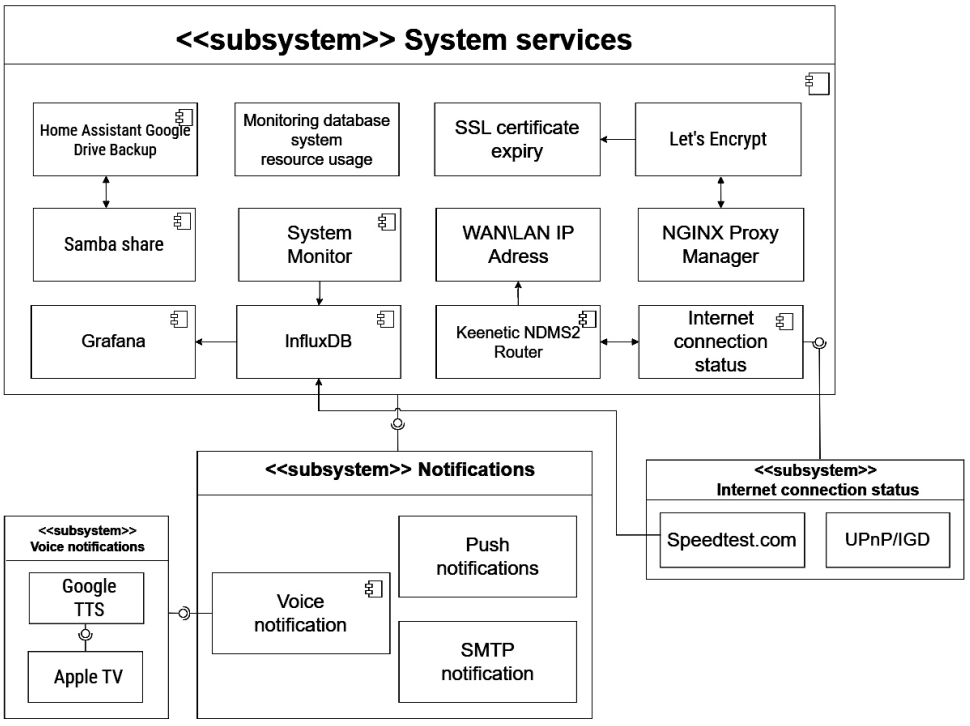

Рис. 2.8: Структура підсистеми «Системні служби» в Home Assistant

Важливим напрямом є моніторинг мережевої взаємодії у локальному середовищі та, за потреби, у зовнішньому доступі. Аналітика щодо активних клієнтів, точок входу, тривалості та частоти з'єднань дає змогу виявляти нетипову активність, що може свідчити про спроби несанкціонованого доступу; таким чином підсистема виконує базові функції мережевого аудиту.

Окремо забезпечується контроль актуальності та збереження системних даних: стану резервних копій, цілісності знімків конфігурації, доступності захищених сховищ. Це гарантує відновлюваність системи у разі збоїв або втрати доступу. Критичною є підтримка криптографічного захисту каналів: автоматизований нагляд за життєвим циклом TLS/SSL-сертифікатів мінімізує ризики переривання сервісів і деградації безпеки.

Модуль сповіщень завершує функціональність підсистеми, забезпечуючи інформування про технічні інциденти, адміністративні події та зміни станів через канали, узгоджені з критичністю (push-повідомлення, електронна пошта, голосові оповіщення). За замовчуванням застосовується підхід *local-first*: зовнішня публікація сервісів (через зворотний проксі) виконується лише за явної потреби та з дотриманням принципу найменших привілеїв.

У підсумку, підсистема «Системні служби» забезпечує операційну стійкість і довготривалу надійність локальної архітектури, підтримуючи узгоджену роботу інших підсистем. Огляд її основних компонент наведено в таблиці 2.5.

Таблиця 2.5: Компоненти та функціональні модулі підсистеми «Системні служби»

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
System Monitor	Моніторинг системних ресурсів (CPU, RAM, диски, навантаження).	Виявлення перевантажень, оптимізація конфігурацій, запуск профілактичних дій.	Передавання телеметрії до InfluxDB; візуалізація у Grafana.
Grafana	Візуалізація даних (графіки, діаграми, дашборди).	Панелі продуктивності: CPU, диски, мережа, доступність інтеграцій.	Працює з InfluxDB як джерелом історичних даних.
InfluxDB	База даних часових рядів для телеметрії.	Зберігання історії моніторингу системи та пристроїв.	Отримує дані від System Monitor та сенсорів Home Assistant; дані споживає Grafana.
Samba Share	Спільний доступ до файлів у локальній мережі.	Резервування конфігурації Home Assistant і службових файлів.	Сумісність із Windows/Linux/Android; ручні та автоматизовані резервні копії.

Продовження таблиці 2.5

Назва компонента	**Функціональне призначення**	**Сценарії використання**	**Взаємодія в системі**
Home Assistant Google Drive Backup*	Автоматичне резервування знімків конфігурації у Google Drive.	Відновлення налаштувань після збоїв, мінімізація втрат даних.	Працює разом із локальним збереженням або Samba; керування через Lovelace.
NGINX Proxy Manager†	Керування зворотним проксі для публікації локальних служб.	Безпечний зовнішній доступ до Home Assistant, Grafana тощо.	Інтеграція з Let's Encrypt для автоматичного оновлення сертифікатів.
Let's Encrypt†	Безкоштовні TLS/SSL-сертифікати з автооновленням.	Захищений HTTPS-доступ до служб у разі зовнішньої публікації.	Працює спільно з NGINX Proxy Manager.
SSL Certificate Expiry	Сенсор контролю строків дії сертифікатів.	Попередження про наближення завершення чинності сертифіката.	Тригери сповіщень і завдань оновлення у зв'язці з Let's Encrypt/NGINX.
WAN/LAN IP Address	Сенсор визначення внутрішньої та зовнішньої IP-адреси.	Контроль мережевого доступу, сценарії зовнішнього доступу, моніторинг змін IP.	Дані з маршрутизатора (наприклад, Keenetic NDMS2) або зовнішніх сервісів.
Keenetic NDMS2 Router	Джерело мережевої аналітики через NDMS2 API.	Стан інтернет-каналу, кількість клієнтів, трафік, зовнішня IP.	Інтеграція з Home Assistant через API; живить датчики WAN/LAN і статусу.
Internet Connection Status	Перевірка доступності зовнішньої мережі.	Сповіщення про втрату зв'язку, перемикання політик логування/резервування.	Реалізація через UPnP, Speedtest або сторонні перевірки доступності.

При цьому слід зазначити, що хмарне резервування опційне й використовується лише за явної потреби, не порушуючи локальну автономність. Зовнішня публікація сервісів допускається тільки за наявності вимоги та з дотриманням принципу найменших привілеїв і сегментації мережі.

2.3. Технічна реалізація, комунікація та захист даних у локальному середовищі автоматизації

Надійна, масштабована та безпечна система автоматизації розумного середовища є неможливою без ретельно спроєктованої мережевої інфраструктури, що забезпечує цілісну інтеграцію гетерогенних пристроїв, різнорівневих каналів зв'язку та функціонально розподілених сервісів управління. В умовах зростання складності цифрових екосистем, зокрема в освітньому та промисловому середовищах, комунікаційний рівень набуває ключового значення як середовище взаємодії між фізичними пристроями та логічними модулями, забезпечуючи необхідний рівень координації, адаптивності та оперативного реагування.

Архітектура запропонованої системи ґрунтується на принципах модульності, децентралізації обчислень і логічної ізоляції компонентів. Такий підхід дозволяє досягти гнучкого масштабування, керованості інформаційних потоків та мінімізації точок відмови. Віртуалізоване середовище, реалізоване на базі гіпервізора *Proxmox VE*, забезпечує високий ступінь абстракції між апаратною та програмною частинами, що спрощує процеси розгортання, оновлення та супроводу. Центральну роль в інфраструктурі відіграє керуючий вузол на базі платформи *Home Assistant*, який виконує функції агрегації подій, реалізації сценарної логіки та забезпечення інтерактивної взаємодії з користувачем у реальному часі. Крім того, Home Assistant інтегрує основні протоколи обміну даними (MQTT, Zigbee2MQTT, REST API), підтримує модульність і сумісність з широким спектром IoT-пристроїв.

Особлива увага в побудові комунікаційної інфраструктури приділена питанням інформаційної безпеки, що мають критичне значення як в умовах часткової автономності системи, так і при її інтеграції з зовнішніми сервісами. Забезпечення надійності комунікацій, цілісності даних і стійкості до атак реалізується шляхом поєднання апаратних і програмних механізмів захисту, інтегрованих на різних рівнях архітектури. Використання наскрізного TLS/SSL-шифрування (включно з автоматичним оновленням сертифікатів через *Let's Encrypt*), сегментації мережі (VLAN, VPN), а також механізмів контролю доступу та аудиту подій забезпечує цифровий суверенітет і знижує ризики компрометації даних чи несанкціонованого доступу.

Таким чином, технічна реалізація локальної системи автоматизації поєднує принципи модульності, віртуалізації та багаторівневого захисту, формуючи надійну інфраструктуру для масштабованих і безпечних IoT-середовищ. У наступних підрозділах буде детально розглянуто практичні аспекти комунікації та механізми захисту даних.

2.3.1. Інфраструктура локального сервера автоматизації.

Архітектура серверного середовища, на якому побудована система моніторингу та автоматизації із застосуванням платформи Home Assistant [126], розроблена відповідно до принципів модульності, функціональної декомпозиції, децентралізації обчислень, протокольної інтероперабельності та відмовостійкості. Такий підхід дозволяє сформувати гнучке середовище, здатне до саморегулювання, масштабування, ізоляції збоїв і забезпечення цілісності обробки даних у реальному часі [141]. Завдяки розділенню логічних і фізичних рівнів система відповідає сучасним вимогам до кіберфізичних систем і систем інформаційного управління.

Центральним компонентом архітектури виступає платформа Home Assistant [142], що функціонує як єдиний інтелектуальний вузол для збору, агрегації, аналізу та керування розподіленими IoT-компонентами. На відміну від централізованих хмарних рішень, Home Assistant реалізує концепцію локальної обробки подій (edge computing) [141], яка забезпечує низьку затримку, мінімальну залежність від зовнішніх сервісів, автономність сценаріїв автоматизації та високий рівень конфіденційності даних.

Фізичний рівень. Формує апаратну основу інфраструктури автоматизації та забезпечує її надійність і безперервність. Реалізований у вигляді автономного серверного вузла, що інтегрує центральний процесор (CPU), модулі оперативної пам'яті (RAM), енергонезалежні накопичувачі (SSD/HDD), мережеві інтерфейси з підтримкою гігабітних/мультигігабітних з'єднань, а також джерела резервного живлення (UPS). Уся система функціонує в межах локальної мережі без обов'язкової інтеграції з хмарними сервісами, що гарантує автономність і повний контроль над даними.

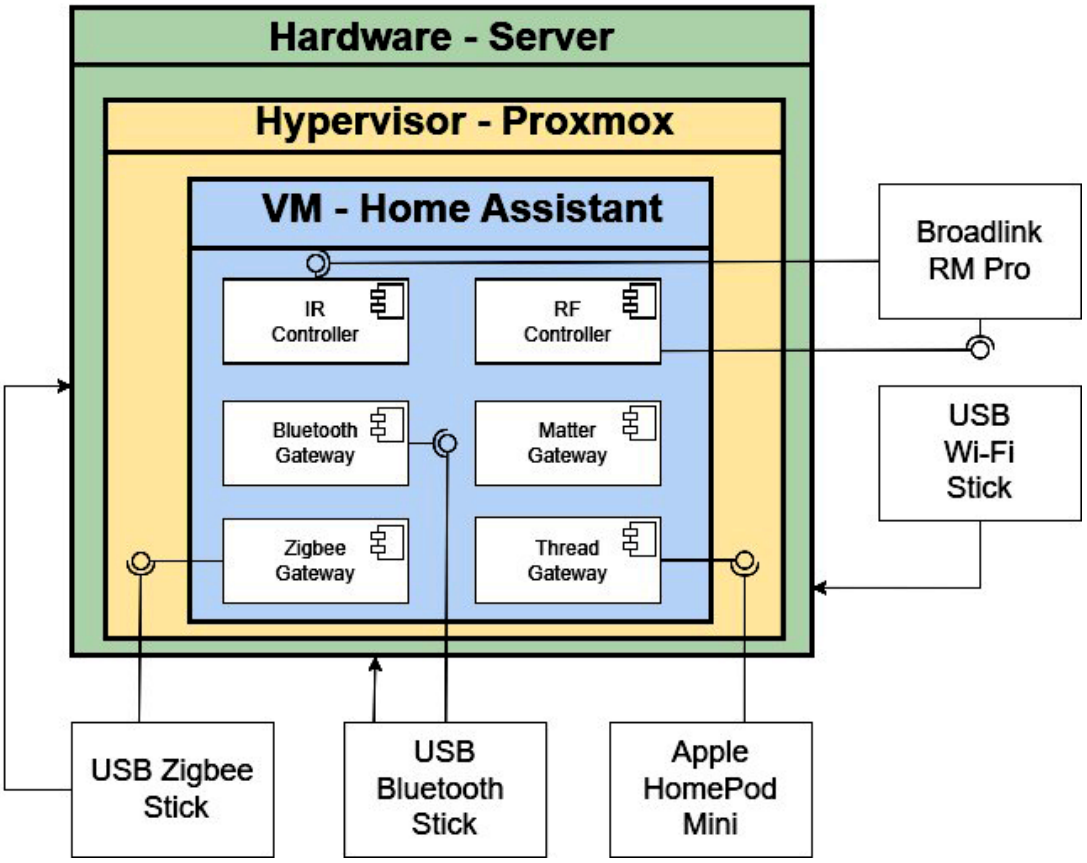

Рис. 2.9: Архітектура серверного середовища системи автоматизації

Гіпервізорний рівень. Реалізований на базі платформи Proxmox Virtual Environment [143], що поєднує гіпервізор KVM [144] і контейнерну технологію LXC [145]. Proxmox забезпечує ізоляцію віртуальних екземплярів компонентів, live migration, кластеризацію з високою доступністю (HA), snapshot-бекапи, REST API та CLI для адміністрування. Контейнери забезпечують ефективність мікросервісної архітектури, у якій окремо функціонують Home Assistant, TimescaleDB, InfluxDB, MQTT-брокер [146], Grafana, Node-RED тощо. Підтримка файлових систем ZFS та BTRFS із механізмами дедуплікації та перевірки цілісності підвищує надійність зберігання. Інтеграція з Prometheus і Proxmox Backup Server посилює стійкість до інфраструктурних збоїв.

Логічний рівень. Реалізується у вигляді віртуальної машини з Home Assistant Operating System [142]. Цей рівень виконує обробку подій, керування пристроями, логіку автоматизації та взаємодію з користувачем. Платформа підтримує понад 2500 інтеграцій, зокрема Zigbee [147], Z-Wave [148], Thread [149], Matter [150], BLE [151], Wi-Fi [152], а також зовнішні сервіси — IFTTT [153], Telegram [154], Weather API [155]. Інформаційна взаємодія здійснюється через MQTT [146], WebSocket [156], REST API [157], TCP/IP [158], що гарантує відкритість до інтеграції з іншими платформами та системами.

Таким чином, трирівнева архітектура інфраструктури локального сервера автоматизації забезпечує збалансоване поєднання продуктивності, відмовостійкості, інформаційної безпеки та масштабованості, що є критично важливим для сучасних IoT-рішень в умовах часткової або повної автономності.

2.3.2. Підключення пристроїв до мережі.

Однією з ключових переваг сучасних платформ для управління розумними середовищами, зокрема Home Assistant [142], є широка підтримка різнорідних протоколів зв'язку та стандартів підключення [146,147,149,150]. Це дозволяє реалізовувати масштабовану, адаптивну та гетерогенну мережеву інфраструктуру, що враховує технічні й експлуатаційні характеристики окремих пристроїв, а також їхнє цільове функціональне призначення.

Архітектурні особливості такої інтеграції демонструються на рисунку 2.10, який відображає топологію

підключення фізичних та віртуалізованих компонентів у межах локального обчислювального середовища.

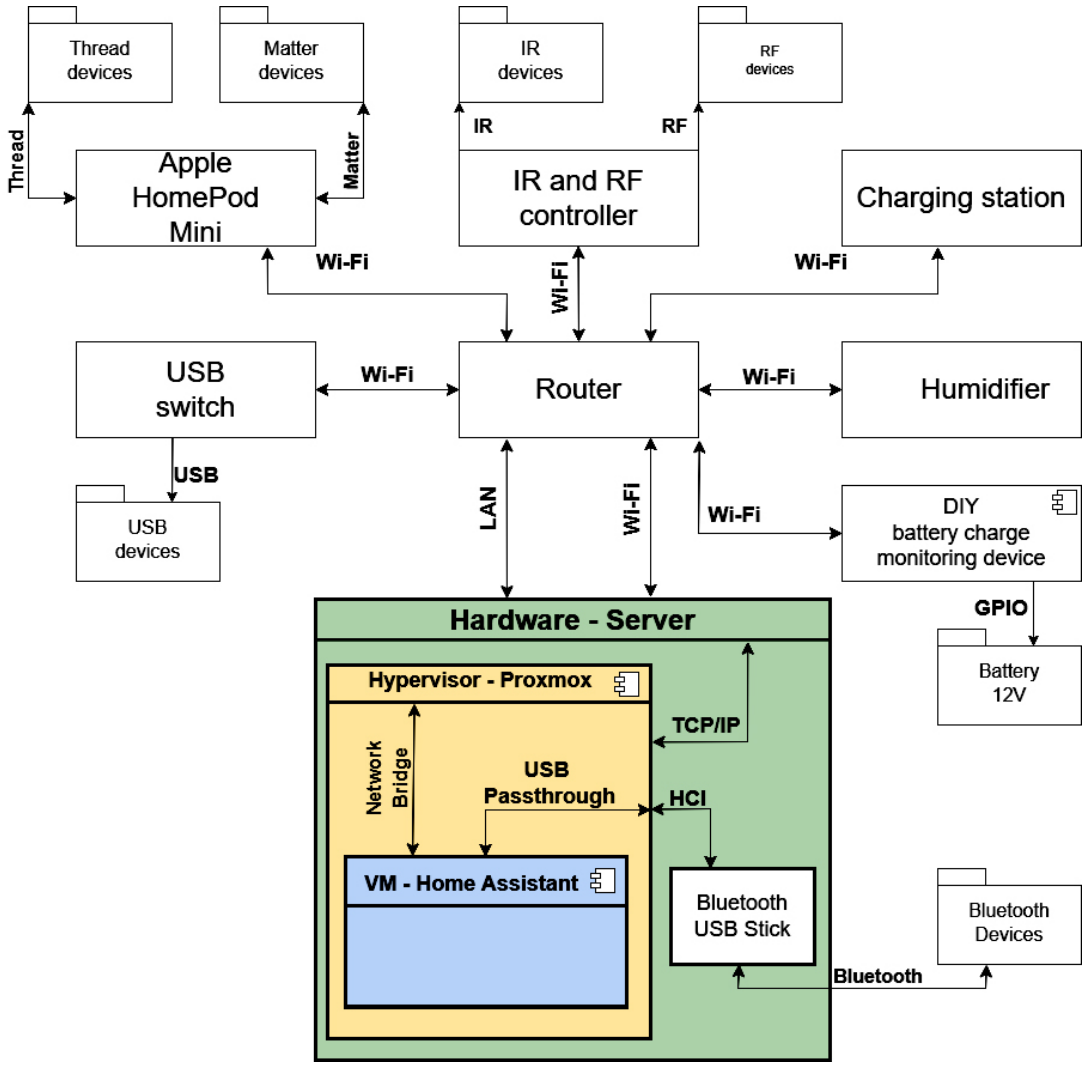

Рис. 2.10: Архітектурна діаграма серверної інфраструктури та протоколів підключення пристроїв

Центральним елементом мережі виступає маршрутизатор, який реалізує як комутацію, так і маршрутизацію трафіку між дротовими (LAN) та бездротовими (Wi-Fi) сегментами [152]. Він забезпечує фізичну та логічну зв'язність між широким спектром пристроїв: керованими модулями побутової автоматизації (зволожувач повітря, зарядна станція), DIY-пристроями на базі мікроконтролерів ESP8266/ESP32 [159], інфрачервоними та RF-контролерами, а також периферійними USB- або Bluetooth-модулями [151].

Обчислювальний вузол представлений сервером з інстальованим гіпервізором Proxmox [143], що забезпечує віртуалізацію сервісів. У його межах функціонує віртуальна машина з Home Assistant [142], яка виконує роль диспетчера автоматизації. Передача даних реалізується через USB passthrough, HCI для Bluetooth або мережевий міст (TCP/IP) для Wi-Fi пристроїв [157].

Окрему роль відіграє Apple HomePod Mini, який виступає шлюзом між Home Assistant та пристроями, що використовують протоколи Thread [149] та Matter [150]. Завдяки цьому забезпечується висока інтероперабельність, незалежно від виробника пристроїв.

Також використовується керований USB-хаб з Wi-Fi підтримкою для гнучкого перемикання периферії між хостами без фізичного втручання - що актуально для лабораторних або тестових конфігурацій. GPIO-інтерфейси в DIY-рішеннях забезпечують прямий контроль за входами/виходами для прикладних задач моніторингу та керування.

Таким чином, побудована мережева інфраструктура поєднує модульність, відкритість стандартів, апаратну абстракцію та віртуалізацію. Це створює основу для надійної, масштабованої, автономної системи управління, орієнтованої на енергоефективність і зниження залежності від зовнішніх хмарних сервісів.

2.3.3. Безпека комунікацій.

Забезпечення безпеки інформаційного обміну в локальній системі автоматизації є ключовим завданням, особливо в умовах її часткової або повної автономності. У межах розробленої архітектури, що базується на платформі Home Assistant [142] у віртуалізованому середовищі Proxmox [143], механізми захисту реалізуються на декількох рівнях - від фізичного сегментування мережі до протокольного шифрування [157, 160] та логічної ізоляції служб.

Комунікаційна інфраструктура побудована за принципом розподіленої сегментації: кожен шлюз, контролер або периферійний пристрій взаємодіє з центральним вузлом через визначений інтерфейс у межах ізольованої внутрішньої мережі (LAN/VLAN). Це дозволяє мінімізувати зовнішній трафік, зменшуючи площину потенційних атак. Усі підключення між компонентами здійснюються через захищені канали з використанням сучасних протоколів - MQTT з TLS-шифруванням [146], HTTPS, WebSocket із токенами доступу [156] та інші стандарти з перевіркою автентичності.

Доступ до веб-інтерфейсу Home Assistant здійснюється виключно через захищене HTTPS-з'єднання з використанням SSL-сертифіката Let's Encrypt, що оновлюється автоматично. Це дозволяє:

- захистити канали зв'язку від перехоплення даних (атаки типу «man-in-the-middle»),
- забезпечити автентичність сервера для клієнтів,
- запобігти підміні веб-інтерфейсу (phishing),
- зменшити ризик несанкціонованого доступу через перехоплення сесій.

Використання відкритого та визнаного сертифікаційного центру Let's Encrypt не лише гарантує сумісність із сучасними браузерами, а й мінімізує витрати на інфраструктуру захисту, при цьому підтримуючи високий рівень довіри.

Постійне оновлення програмного середовища є важливою складовою безпеки. Платформи Proxmox і Home Assistant мають активні спільноти розробників та регулярно отримують оновлення, що усувають відомі вразливості, покращують механізми автентифікації, захисту API та внутрішньої ізоляції. Завдяки цьому забезпечується актуальність компонентів, а також оперативна реакція на нові кіберзагрози, що знижує ризики експлуатації вразливостей у системі.

У системі реалізовано багаторівневу модель доступу з вбудованою авторизацією як у Proxmox, так і у Home Assistant.

У Proxmox VE підтримується розмежування прав на рівні користувачів, груп і ролей. До послуг адміністратора - повноцінний контроль доступу на основі RBAC (Role-Based Access Control), який дозволяє:

- створювати користувачів з різним рівнем доступу (наприклад, лише до перегляду журналів або повного адміністрування),
- обмежувати дії в межах окремих віртуальних машин, контейнерів або вузлів,
- використовувати LDAP або двофакторну автентифікацію (TOTP) для підвищеного захисту.

Home Assistant також має власну систему керування користувачами, яка включає:

- облікові записи з розподіленням ролей (адміністратор, користувач),
- вбудовану підтримку 2FA (двохфакторної автентифікації) за допомогою мобільного застосунку або генератора кодів,
- захист від підбору паролів за рахунок затримок при повторних спробах входу,
- журнал доступу з фіксацією IP-адрес, часу входу та повідомленнями про неуспішні спроби.

Система надсилає сповіщення про нові входи та невдалі спроби авторизації, що дозволяє оперативно реагувати на потенційні загрози. Усе це значно підвищує загальний рівень захищеності інтерфейсів адміністрування та управління.

Завдяки повній локалізації обробки подій Home Assistant не залежить від хмарних сервісів, що виключає неконтрольований обіг даних поза межами приватного середовища. Це підвищує конфіденційність і забезпечує

контроль над критичними потоками інформації. Віртуальні машини ізольовано виконують службові ролі, а USB passthrough дозволяє точно контролювати доступ фізичних інтерфейсів до окремих сервісів без втрати гнучкості в підключенні пристроїв [161].

Особливу увагу приділено безпечному відеоспостереженню. Потоки з камер отримуються напряму за протоколами RTSP або ONVIF [162] без участі зовнішніх серверів. Обробка відео - детекція руху, ідентифікація об'єктів - виконується на локальному сервері за допомогою програмних рішень на зразок Frigate або MotionEye [163]. Усі відеодані зберігаються на внутрішніх накопичувачах, що унеможливлює зовнішній доступ до зображень і метаданих, а також забезпечує відповідність вимогам до збереження приватності. На рівні мережевої аналітики система використовує моніторингові сервіси Prometheus, Grafana, Loki [164, 165], які дозволяють відстежувати з'єднання, ідентифікувати аномалії трафіку, фіксувати підозрілі активності (наприклад, сканування портів або несанкціоновані спроби підключення). Це дає змогу оперативно застосовувати превентивні політики безпеки без шкоди для продуктивності системи.

Таким чином, комунікаційна модель не обмежується функціональним з'єднанням між компонентами, а виступає цілісною системою інформаційного захисту. Локалізоване обчислення, структурна ізоляція, зашифровані канали та контрольований доступ формують надійну платформу, що відповідає сучасним викликам кібербезпеки в сфері автоматизації.

2.4. Висновки по розділу 2

1. Представлено результати дослідження, спрямованого на проєктування архітектури, функціональної організації та технічної реалізації локальної IoT-системи моніторингу та керування умовами приміщень на базі платформи Home Assistant. Запропонована модель охоплює всі ключові рівні — від фізичного (датчики, актуатори) до логічного (сценарії, аналітика, інтерфейс) — та забезпечує автономну роботу без залежності від зовнішніх хмарних сервісів, гарантуючи високий рівень безпеки, гнучкості й масштабованості.
2. Запропоновано комплексну багаторівневу архітектуру локальної IoT-системи, побудованої на платформі Home Assistant, яка охоплює сім взаємопов'язаних функціональних рівнів: пристроїв, комунікацій, даних, керування, безпеки, сервісів та презентації. Така модульна структура забезпечує незалежність від зовнішніх інфраструктур, підвищену стійкість до змін у мережевому оточенні та можливість гнучкого масштабування без втрати керованості й функціональної цілісності системи.
3. Розроблено методику багатокритеріальної оцінки ефективності локальної IoT-системи, яка передбачає інтеграцію часткових показників продуктивності, надійності, інформаційної безпеки та енергоспоживання в узагальнений критерій ефективності. Це дозволяє здійснювати об'єктивне порівняння різних конфігурацій системи, виявляти «вузькі місця» в архітектурі та формувати оптимальні сценарії подальшої модернізації.
4. Набули подальшого розвитку теоретичні основи побудови багаторівневої IoT-архітектури, зокрема у частині математичної формалізації взаємодії компонентів локальної системи. Розроблена модель дозволяє формалізувати інформаційні потоки та взаємозв'язки між компонентами на кожному з рівнів, що є підґрунтям для подальшої оптимізації та автоматизації сценаріїв керування.
5. Удосконалено підхід до реалізації механізмів кібербезпеки в локальних IoT-системах шляхом наскрізної інтеграції компонентів контролю доступу, сегментації мережі, TLS/SSL-шифрування, систем журналювання активності та підтримки багатофакторної автентифікації (2FA). Комплексний рівневий захист охоплює як фізичні, так і логічні вузли системи, що дозволяє мінімізувати ризики витоку даних, зловмисного втручання та забезпечити цифровий суверенітет користувача відповідно до сучасних стандартів інформаційної безпеки.
6. Удосконалено методи керування пристроями та сценаріями автоматизації шляхом впровадження шаблонізованих конфігурацій, тригерно-орієнтованої логіки, умовних операторів і розділення рівнів прийняття рішень (система – користувач – пристрій). Такий підхід дозволяє створювати динамічні сценарії

поведінки системи, які реагують на зміну стану середовища або взаємодію з користувачем без необхідності ручного редагування конфігураційних файлів чи перезапуску служб, що значно підвищує стабільність, керованість і адаптивність системи в експлуатації.

7. Сформовано сервісний рівень взаємодії у вигляді уніфікованого API-шару, який підтримує двосторонню асинхронну комунікацію в реальному часі з використанням REST та WebSocket протоколів. Це забезпечує інтеграцію системи з зовнішніми програмними модулями та службами, гнучке масштабування функціоналу та можливість побудови розподілених обчислювальних рішень без втрати цілісності даних, продуктивності та безпеки.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ ЛОКАЛЬНОЇ IOT-СИСТЕМИ НА ОСНОВІ HOME ASSISTANT

У цьому розділі представлені результати практичної реалізації локальної IoT-системи на основі платформи *Home Assistant*, що була концептуально спроєктована та формалізована у другому розділі. Основна увага приділяється переходу від теоретичних рішень до їхньої безпосередньої імплементації у програмному та апаратному середовищі з подальшою перевіркою відповідності заявленим у вступі дисертації вимогам.

Спочатку представлено загальний інтерфейс системи, який забезпечує інтерактивну взаємодію користувача з функціональними підсистемами. Цей інтерфейс є ключовим елементом користувацького досвіду, оскільки дозволяє здійснювати моніторинг стану приміщень, оперативне керування пристроями та налаштування сценаріїв автоматизації без необхідності використання зовнішніх сервісів.

Подальший виклад присвячено поетапній реалізації компонентів системи. Детально розглянуто підсистеми безпеки, кліматичного контролю, освітлення, енергоспоживання та сервісних модулів, а також особливості їхньої інтеграції у єдине середовище. Окремо проаналізовано організацію комунікаційних каналів та механізми захисту даних у локальному середовищі, що гарантують автономність і підвищений рівень інформаційної безпеки.

Основні результати, представлені у цьому розділі, опубліковані у роботах автора [127, 129]

3.1. Інтерфейс головної сторінки системи автоматизації

3.1.1. Загальна концепція та структура.

Головна сторінка інтерфейсу локальної системи автоматизації, реалізованої у середовищі *Home Assistant*, побудована на принципах модульності, інформативності та адаптивної візуалізації. Інтерфейс забезпечує централізований доступ до показників усіх підсистем, підтримує оперативний моніторинг та надає засоби керування виконавчими пристроями в режимі реального часу.

Інформація подається у вигляді дашбордів, які виступають незалежними функціональними блоками та можуть містити сенсорні індикатори, графіки часових рядів, кнопки керування або статусні віджети. Такий підхід дозволяє створити логічно впорядковане й інтуїтивне середовище для взаємодії з системою. На рисунку 3.1 подано приклад основної сторінки, що демонструє організацію дашбордів та структуру подання зведених даних.

На головній сторінці інтегровано віджети кліматичного моніторингу, інформацію щодо якості повітря, показники енергоспоживання, елементи системи безпеки, діагностичні повідомлення, а також інструменти взаємодії з сервісними модулями. Завдяки миттєвому оновленню даних кожна зміна стану пристроїв негайно відображається в інтерфейсі, що забезпечує точний зворотний зв'язок у реальному часі.

Структура інтерфейсу підтримує адаптивний дизайн, включаючи темний режим, ієрархію елементів, кольорове кодування станів та використання піктограм. Технічно інтерфейс реалізовано за допомогою *Lovelace UI*, що підтримує як YAML-конфігурацію, так і інтерактивне редагування. Розміщення блоків може бути динамічно змінене відповідно до сценарію використання або ролі користувача.

Таким чином, головна сторінка виконує функції інформаційного центру та координуючого вузла системи автоматизації. Вона забезпечує інтеграцію всіх підсистем у єдиному візуальному середовищі, поєднуючи моніторинг, аналітику та керування у структурованому та зручному форматі.

3.1.2. Бічне меню навігації.

У лівій частині інтерфейсу *Home Assistant* розташовано бічне меню навігації, яке забезпечує оперативний доступ до основних модулів системи. Меню структуроване логічно та охоплює як модулі моніторингу, так і

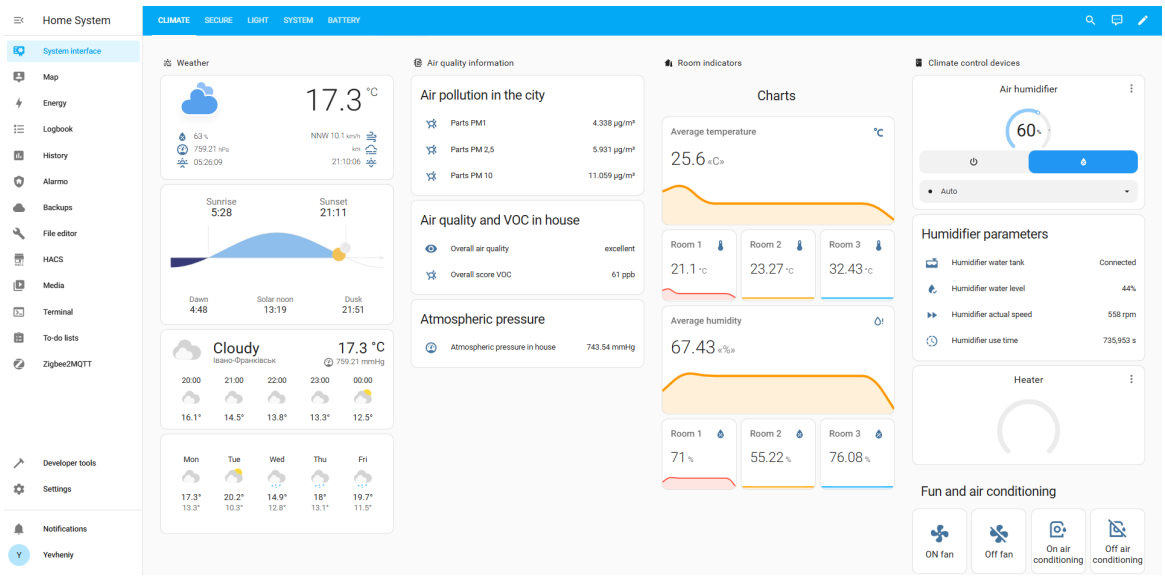

Рис. 3.1: Головна сторінка інтерфейсу Home Assistant із відображенням основних підсистем та інтерактивних дашбордів.

засоби адміністрування, діагностики й керування автоматизацією. У цьому підрозділі наведено опис ключових інструментів навігаційного меню.

3.1.2.1. Моніторинг та аналітика середовища.

Модулі моніторингу та аналітики забезпечують інтегроване подання просторових і часових параметрів системи, а також створюють основу для контекстно-залежних сценаріїв автоматизації.

Модуль «Карта» (рисунок 3.2) забезпечує візуалізацію просторового розташування об'єктів системи на основі географічних координат. Він дозволяє відображати у режимі реального часу місцезнаходження користувачів, мобільних пристроїв, транспортних засобів або трекерів, що передають GPS-дані через застосунки *Home Assistant Companion*, зовнішні API або спеціалізовані сенсори.

Інформація про геопозицію може бути використана як тригер у сценаріях автоматизації. Наприклад, вхід користувача до визначеної геозони може ініціювати увімкнення освітлення, активацію кліматичного контролю або зміну стану охоронної системи. Вихід із геозони, своєю чергою, може запускати енергозберігальні режими або коригувати температурні параметри.

Модуль підтримує багатокористувацький режим, динамічне оновлення координат та збереження історії руху (треків) для подальшого аналітичного опрацювання. Візуалізація реалізована на базі *OpenStreetMap/Leaflet*, що забезпечує масштабованість і незалежність від комерційних картографічних сервісів.

Модуль «Енергоменеджмент» призначений для моніторингу та аналізу енергоспоживання в межах локальної автоматизованої системи. Він підтримує інтеграцію з цифровими лічильниками (PZEM, Shelly EM), інверторами відновлюваної енергетики, контролерами навантаження, а також сервісами визначення тарифів (наприклад, *Electricity Maps*).

Панель відображає як загальний енергетичний баланс системи, так і детальний розподіл споживання за зонами чи окремими пристроями (рисунки 3.3–3.4). У режимі реального часу можна відстежувати взаємодію між джерелами живлення (мережею, сонячними панелями, акумуляторними батареями) та споживачами, що подано у вигляді діаграми потоків енергії.

Модуль може використовуватися як аналітичний інструмент для побудови автоматизацій, що враховують тарифні сітки, графіки споживання, потужність джерел та поведінкові патерни користувачів. Це дозволяє оптимізувати енергоспоживання й підвищити ефективність використання відновлюваних джерел.

Модуль «Журнал подій» (рисунок 3.5) забезпечує хронологічний аудит системи, фіксуючи зміни станів пристроїв, активації тригерів, запуск автоматизацій та дії користувачів. Для кожної події зберігається час,

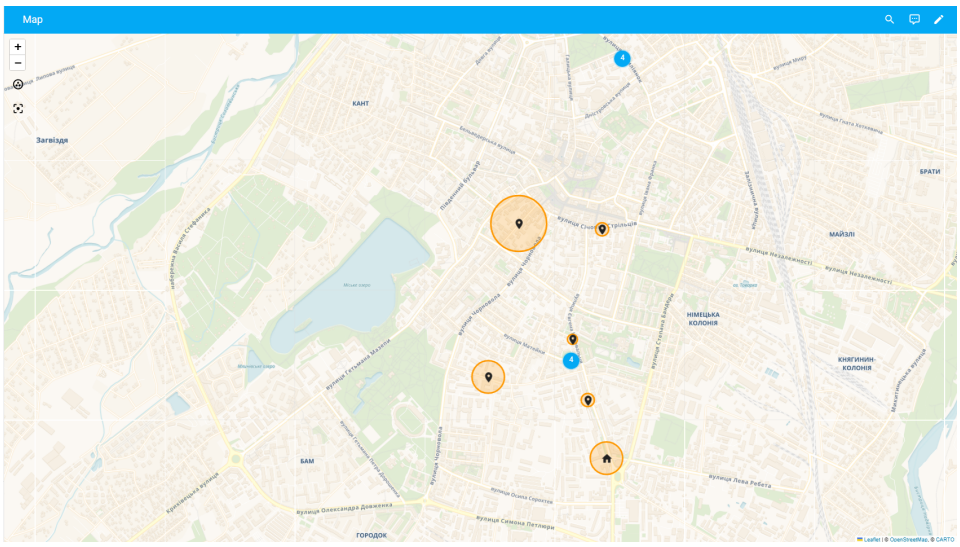

Рис. 3.2: Інтерфейс модуля «Карта» з відображенням геопозиції користувача

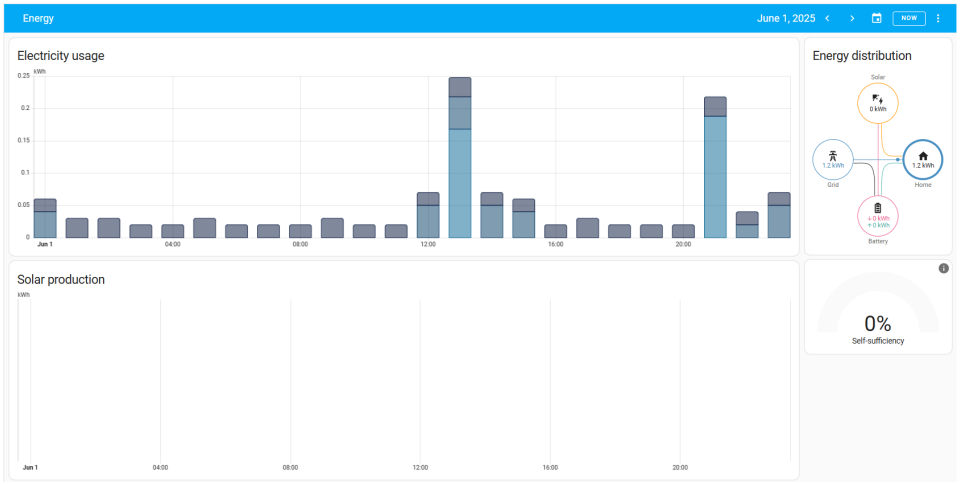

Рис. 3.3: Загальна візуалізація балансу споживання та виробництва електроенергії

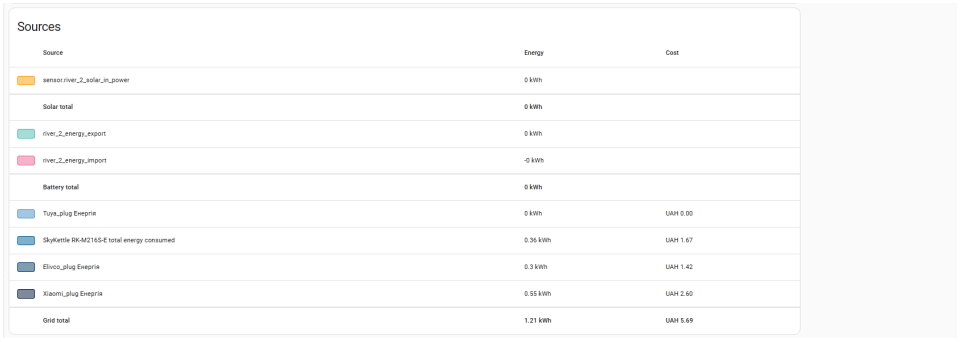

Рис. 3.4: Розгорнутий перелік споживачів із показниками енергії та вартості

ідентифікатор сутності, попередній та поточний стани, а також спосіб ініціації (автоматично, вручну або через API).

Модуль є важливим інструментом під час налагодження системи, діагностики інцидентів і верифікації автоматизацій. Засоби фільтрації та пошуку спрощують аналітичну роботу й забезпечують прозорість усіх процесів.

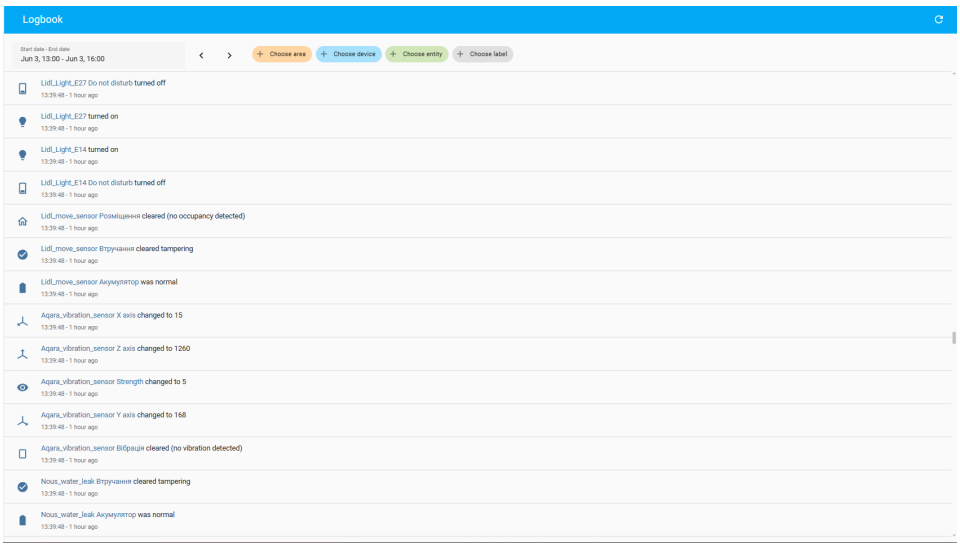

Рис. 3.5: Інтерфейс модулю «Журнал подій» у середовищі Home Assistant

Модуль «Історія» (рисунок 3.6) призначений для аналізу динаміки параметрів системи у часовому вимірі. Він дозволяє переглядати графіки зміни значень сенсорів та станів виконавчих пристроїв із можливістю вибору періоду, порівняння кількох часових серій та фільтрації даних.

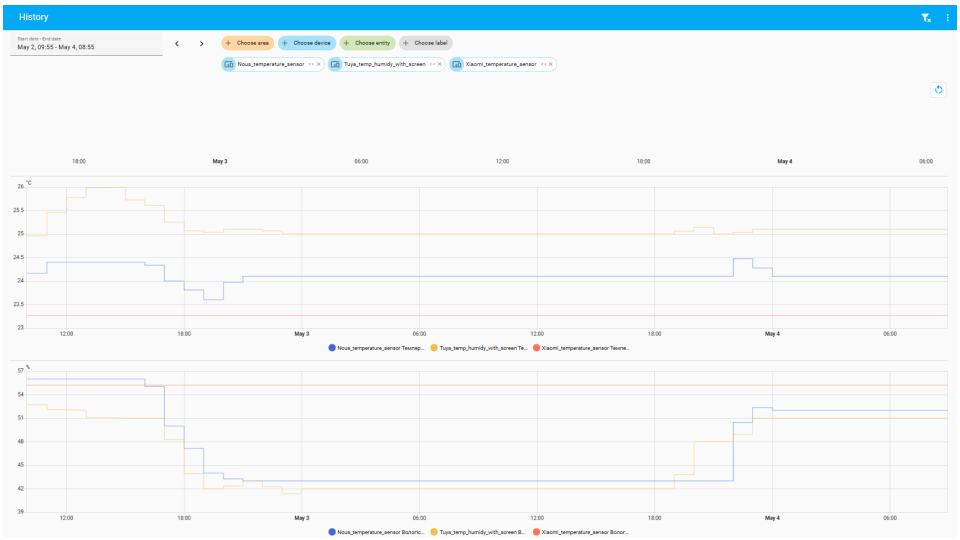

Рис. 3.6: Інтерфейс модуля «Історія»

Дані можуть зберігатися у спеціалізованих часових базах даних, таких як *TimescaleDB* або *InfluxDB*. Це забезпечує ефективну роботу з великими обсягами часових рядів, що є необхідним для побудови прогнозів, ретроспективного аналізу та оцінки стабільності системи у довгостроковій перспективі.

3.1.3. Керування підсистемами та автоматизацією.

Функціональні модулі керування підсистемами забезпечують інтегроване управління безпековими, енергетичними, сервісними та комунікаційними компонентами системи автоматизації. Нижче наведено опис ключових інструментів, що реалізують ці можливості.

Модуль *Alarmo* реалізує підсистему охоронної сигналізації з багатозонним моніторингом, сценарним керуванням та контекстно-залежними реакціями. Підтримуються режими повної, часткової та нічної охорони, налаштування затримок на вхід/вихід, часові обмеження та диференційовані рівні доступу користувачів.

Розділ General (Загальні налаштування) (рисунок 3.7) визначає базові параметри функціонування системи: доступні режими охорони, затримки активації та деактивації, тривалість тривожного стану, а також дозволені методи керування. Ці параметри формують основу логіки безпекових сценаріїв та дають змогу адаптувати поведінку системи до вимог користувачів і умов середовища.

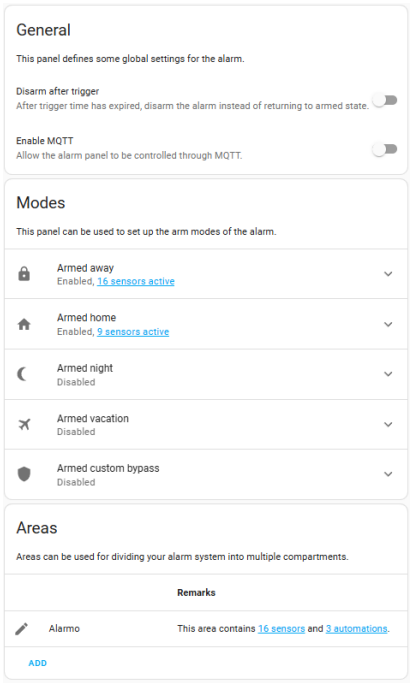

Рис. 3.7: Налаштування Alarmo — розділ «General»

У розділі Sensors (Датчики) (рисунок 3.8) налаштовується зв'язок системи з фізичними сенсорами, що фіксують рух, відкриття дверей і вікон, розбиття скла або інші події. Для кожного сенсора задаються тип тригера, активність у різних режимах охорони та зональна приналежність, що дозволяє створювати багаторівневу, зоноорієнтовану архітектуру охорони.

Розділ Codes (Коди доступу) (рисунок 3.9) забезпечує створення та керування персоналізованими PIN-кодами для авторизованого доступу до системи. Для кожного коду можуть задаватися індивідуальні права, часові обмеження та область дії, що дає змогу гнучко розмежовувати повноваження різних користувачів або груп.

У розділі Actions (Дії у відповідь) (рисунок 3.10) налаштовуються сценарії реакцій системи на події тривоги. До них належать звукові та світлові сповіщення, активація сирен, запис відео з камер, автоматичне вмикання освітлення та надсилання повідомлень користувачам. Реакції можуть визначатися з урахуванням контексту — часу доби, активного режиму охорони або присутності користувачів.

Інтерактивна панель керування (рисунок 3.11) відображає поточний статус системи, активні сенсори, таймери затримки та обраний режим охорони. Завдяки інтеграції з основною інфраструктурою *Home Assistant* модуль *Alarmo* може взаємодіяти з іншими підсистемами (освітлення, клімат, відеоспостереження), формуючи єдиний комплекс автоматизації функцій безпеки.

3.1.3.1. Zigbee2MQTT..

Модуль *Zigbee2MQTT* реалізує повнофункціональний інтерфейс для керування мережею пристроїв стандарту Zigbee через MQTT-шлюз, забезпечуючи прозору комунікацію між сенсорами, виконавчими пристроями та серверною частиною системи *Home Assistant*. Архітектура інтеграції побудована за принципом публікації–підписки (publish–subscribe), що забезпечує масштабованість, сумісність і ефективну обробку подій у режимі реального часу. Користувацький інтерфейс організовано у вигляді вкладок, кожна з яких виконує окрему функціональну роль у процесах моніторингу, конфігурації та аналітики мережі Zigbee.

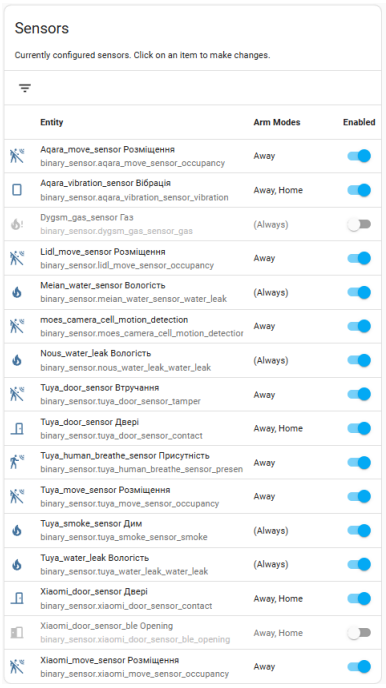

Рис. 3.8: Налаштування Alarmo — розділ «Sensors»

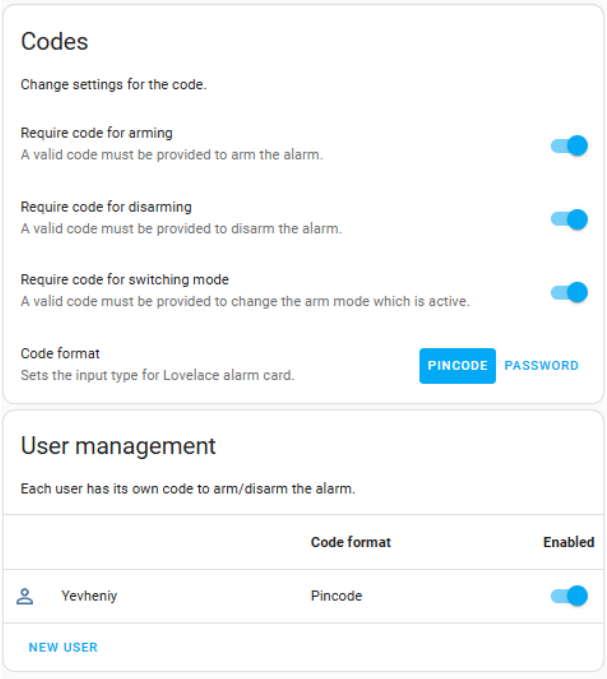

Рис. 3.9: Налаштування Alarmo — розділ «Codes»

Вкладка «Devices» (рисунок 3.12) містить перелік підключених пристроїв із відображенням основних технічних параметрів: типу, виробника, рівня сигналу (LQI/RSSI), версії прошивки, часу останньої активності та доступних команд керування. Цей розділ забезпечує базову ідентифікацію вузлів мережі та дає змогу виконувати оперативну діагностику їхнього стану.

Вкладка «Dashboard» (рисунок 3.13) забезпечує динамічну візуалізацію показників мережі в режимі реального часу, зокрема активності вузлів, рівня сигналу, затримок передавання даних та загального стану шлюзу. Панель використовується для поточного моніторингу продуктивності й стабільності комунікацій.

Вкладка «Map» (рисунок 3.14) відображає топологію Zigbee-мережі у вигляді графічної схеми з візуалі-

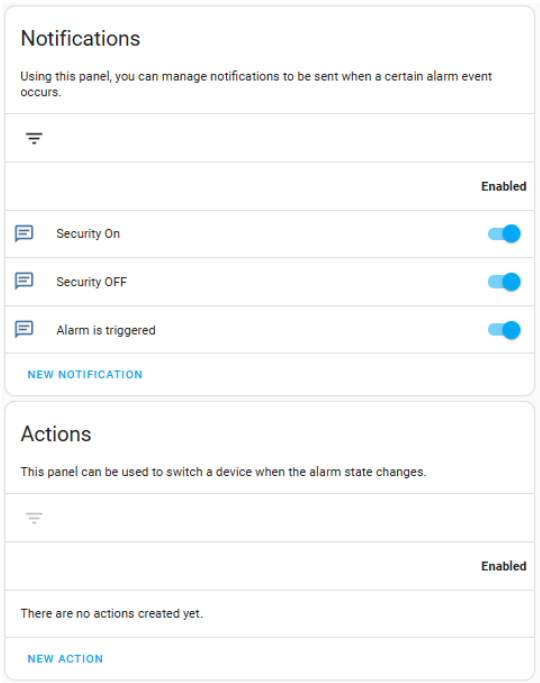

Рис. 3.10: Налаштування Alarmo — розділ «Actions»

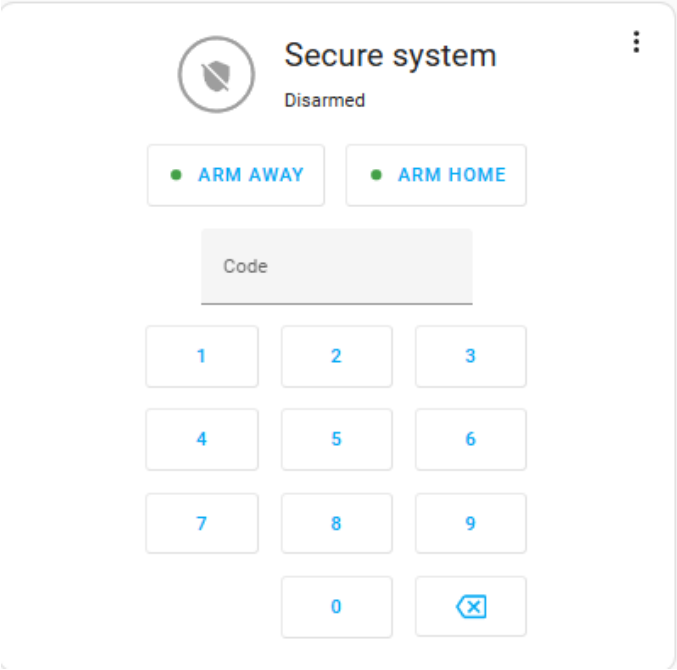

Рис. 3.11: Панель керування Alarmo

зацією маршрутів зв'язку та якості з'єднань між вузлами. Це дає змогу виявляти проблемні ділянки мережі, аналізувати ефективність маршрутизації та оптимізувати розташування пристроїв для підвищення надійності передавання даних.

Додаткові вкладки (Groups, OTA, Touchlink, Logs, Extensions) забезпечують розширені можливості керування та обслуговування мережі:

- Groups — формування логічних груп пристроїв для колективного керування та синхронізованих дій;
- OTA — централізоване оновлення прошивок пристроїв «по повітрю» (рисунок 3.15);
- Touchlink — швидке додавання сумісних пристроїв через безпосередній контакт;

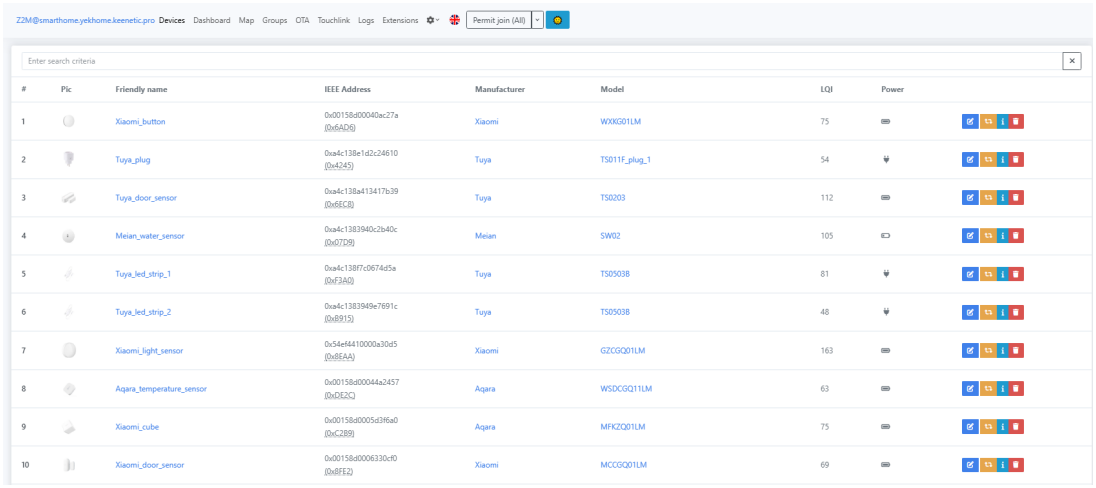

Рис. 3.12: Вкладка «Devices» інтерфейсу Zigbee2MQTT

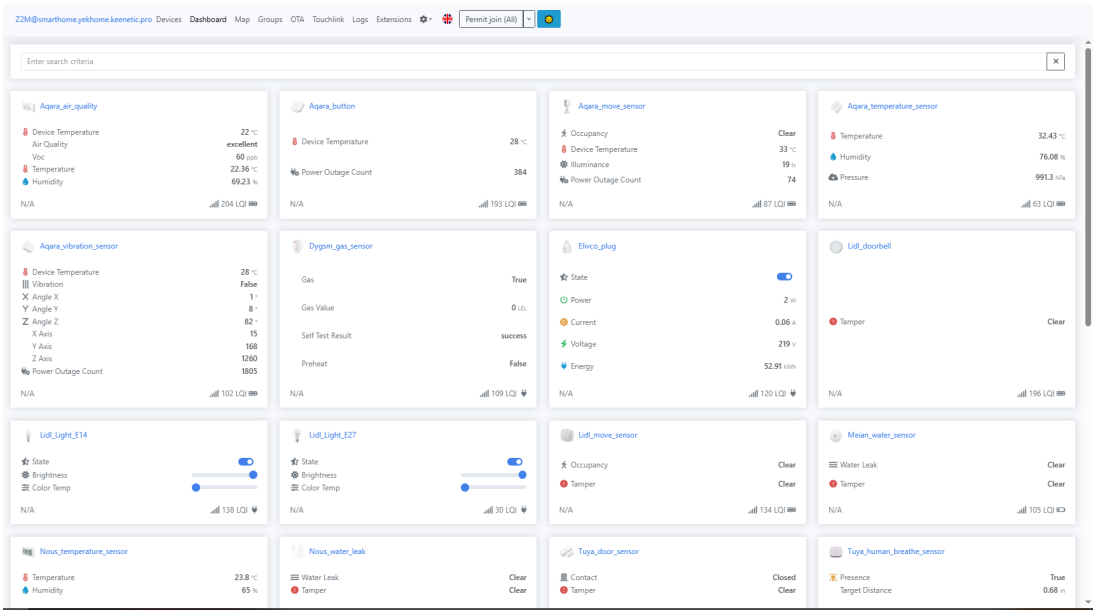

Рис. 3.13: Вкладка «Dashboard» інтерфейсу Zigbee2MQTT

- Logs — детальне журналювання системних подій і діагностичних повідомлень (рисунок 3.16);
- Extensions — підключення зовнішніх модулів і розширень для збільшення функціональності шлюзу.

Таким чином, модуль *Zigbee2MQTT* є ключовим компонентом комунікаційної підсистеми розробленої системи автоматизації, забезпечуючи її гнучкість, розширюваність та незалежність від пропрієтарних рішень.

3.1.3.2. Модуль Медіа (Media).

Модуль «Медіа» забезпечує централізоване керування мультимедійним контентом у межах розробленої системи автоматизації. Вона інтегрує локальні сховища та хмарні стримінгові сервіси, такі як *Spotify*, *YouTube*, *TuneIn*, *Text-to-Speech (TTS)*, *UPnP/DLNA*, і забезпечує відтворення аудіо- та відеопотоків на сумісних пристроях — мультирум-системах, *Google Nest*, *Amazon Echo*, *Smart TV* тощо. Архітектура підсистеми орієнтована на уніфіковане керування потоковими медіаданими з можливістю синхронізації відтворення між кількома пристроями.

Функціонал модуля охоплює:

- централізоване відтворення аудіо та відео на різних пристроях з підтримкою синхронізації (*multi-room streaming*);

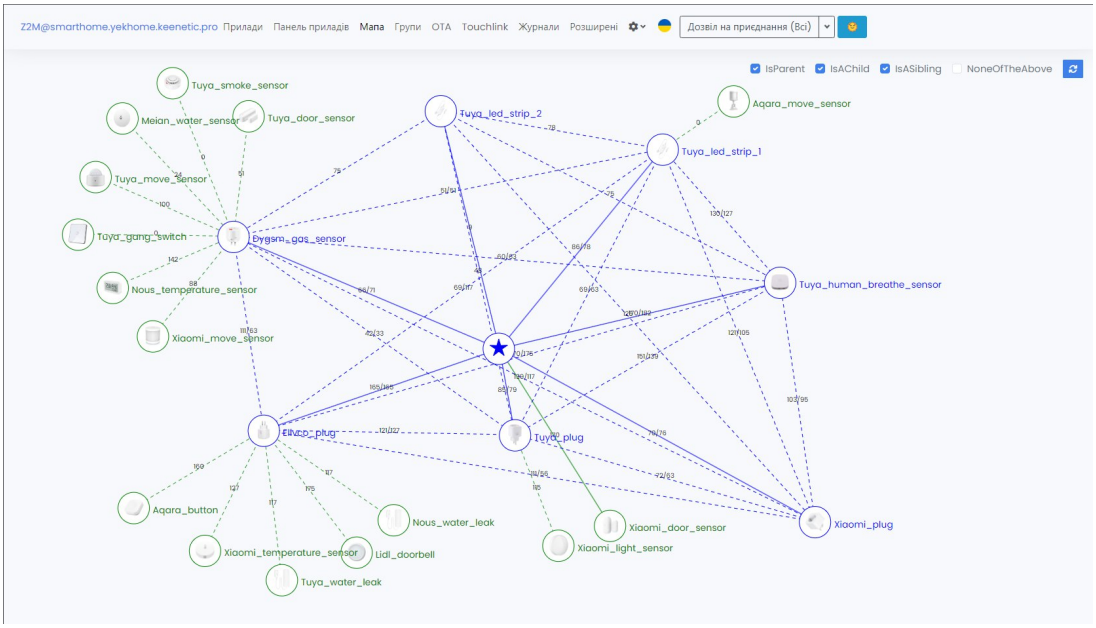

Рис. 3.14: Вкладка «Map» інтерфейсу Zigbee2MQTT

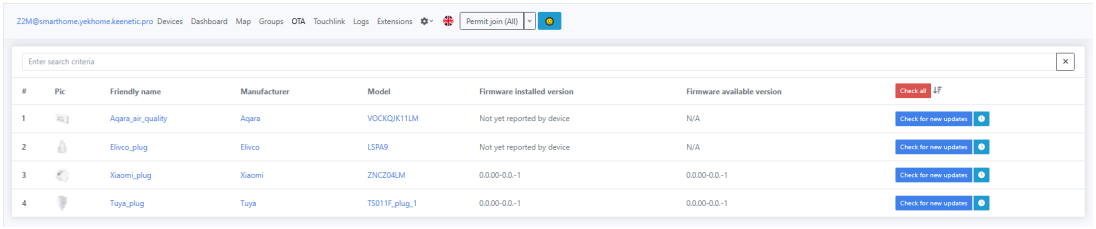

Рис. 3.15: Вкладка «OTA» інтерфейсу Zigbee2MQTT

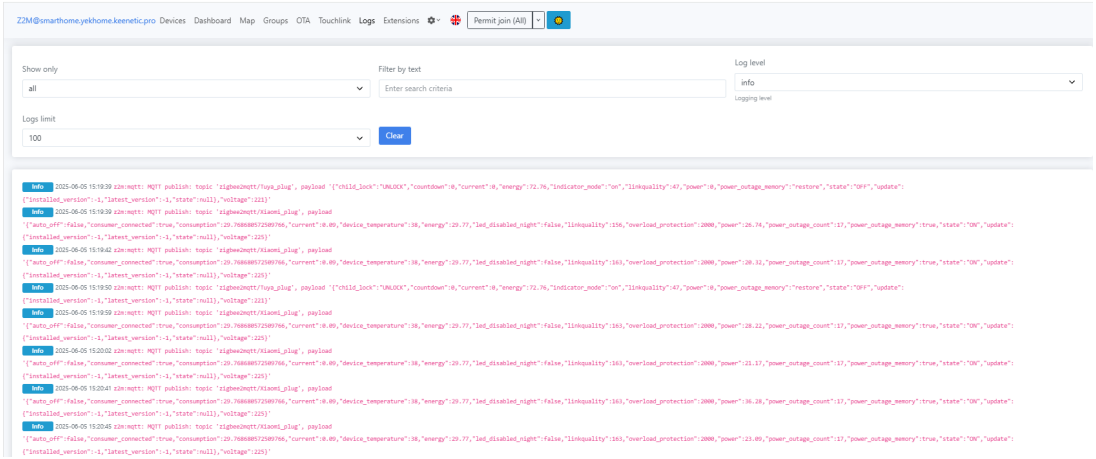

Рис. 3.16: Вкладка «Logs» інтерфейсу Zigbee2MQTT

- керування чергами відтворення, перемикання треків та динамічний вибір джерела контенту;
- регулювання гучності та маршрутизацію аудіосигналу на вибрані пристрої;
- інтеграцію з сервісами *Text-to-Speech (TTS)* для голосових повідомлень та автоматичних сповіщень.

Інтерфейс керування може бути реалізований як окрема вкладка або інтерактивний віджет на дашборді, що забезпечує швидкий доступ до основних функцій: вибору джерела, контролю відтворення, регулювання гучності та зміни аудіовиходу. Завдяки гнучкості налаштувань підсистема може використовуватися у сценаріях

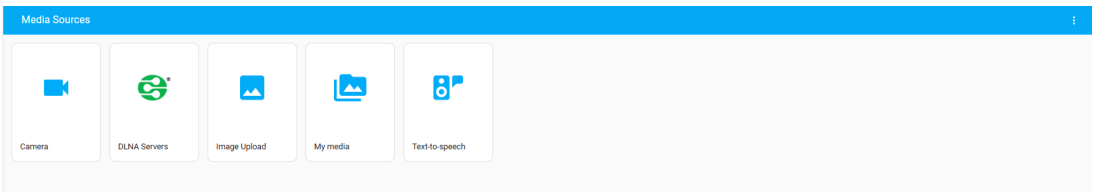

Рис. 3.17: Інтерфейс модуля керування медіаконтентом

автоматизації, наприклад, для відтворення інформаційних повідомлень, аудіоінструкцій або фонової музики під час певних подій.

3.1.3.3. Модуль «Список справ» (To-do List).

Модуль «Список справ» (*To-do List*) виконує роль інтерактивного інструмента для планування, організації та моніторингу завдань у межах системи автоматизації. Він підтримує багаторівневу структуру переліків, встановлення термінів виконання, змінні статуси завдань, додавання коментарів і позначок пріоритетів. Інтерфейс модуля подано на рисунку 3.18.

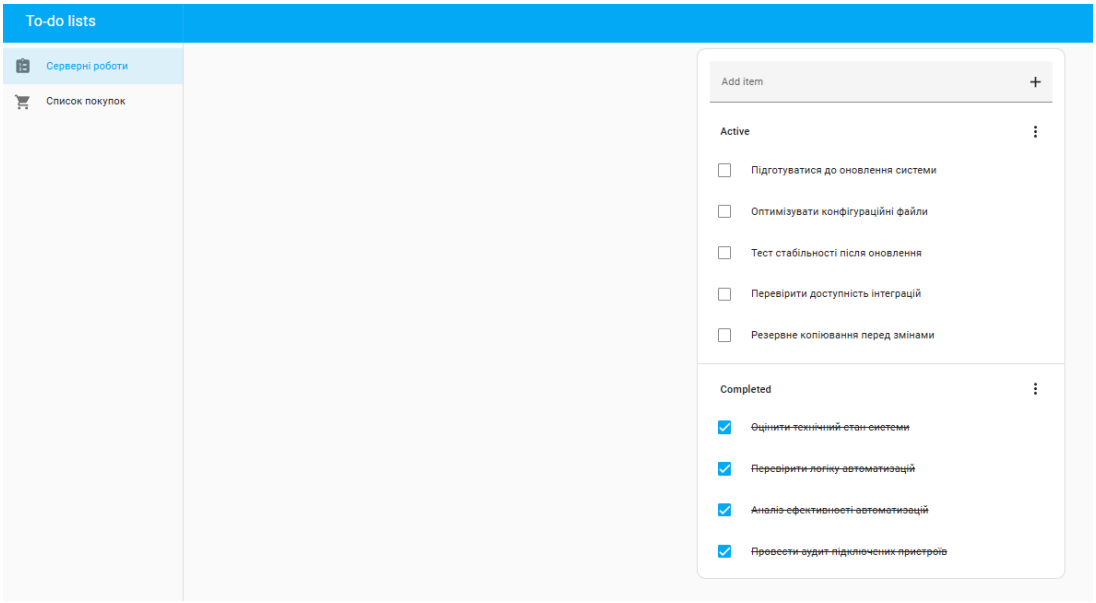

Рис. 3.18: Модуль «To-do List» у середовищі Home Assistant

Особливістю модуля є інтеграція з підсистемою автоматизації, що дає змогу створювати нові записи на основі подій системи або значень сенсорів. Наприклад, при досягненні порогового значення температури, вологості чи рівня освітлення може автоматично створюватися завдання на перевірку відповідного пристрою або ініціацію технічного обслуговування. Відмітка про виконання завдання, своєю чергою, може бути використана як тригер для запуску сценарію (вимкнення пристрою, оновлення журналу подій, надсилання сповіщення тощо).

Таким чином, модуль «To-do List» розширює функціональні можливості системи, забезпечуючи взаємозв'язок між керуванням завданнями та автоматизованими процесами та підвищуючи рівень автономності й адаптивності розробленого середовища.

3.1.3.4. Адміністрування та технічне обслуговування.

Підсистема адміністрування забезпечує підтримку життєвого циклу системи автоматизації, включаючи резервне копіювання, редагування конфігурацій, управління розширеннями та доступ до діагностичних інструментів. Її функціональність спрямована на підтримання стабільності, безпеки та розширюваності середовища *Home Assistant*.

Модуль «Backups» забезпечує створення, зберігання та відновлення повних або часткових резервних копій налаштувань, інтеграцій, баз даних, YAML-файлів і панелей інтерфейсу. Підтримується як локальне, так і хмарне зберігання копій (через *Google Drive*, *Dropbox*, *WebDAV*) із можливістю шифрування, автоматизації процесів і реалізацією політики ротації резервів. Інтерфейс модуля наведено на рисунку 3.19.

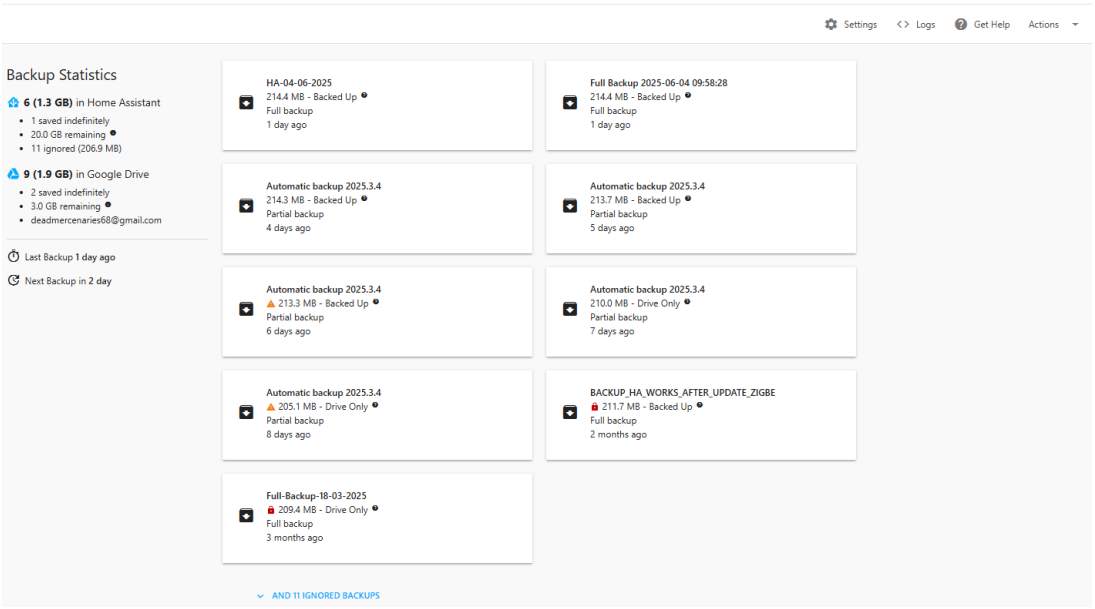

Рис. 3.19: Модуль резервного копіювання: список копій та опції відновлення

Модуль «File Editor» є інтегрованим середовищем для редагування конфігураційних файлів YAML. Він підтримує підсвітку синтаксису, пошук, створення, перейменування та видалення файлів і тек, автозбереження, а також інтеграцію з *Developer Tools* для перевірки коректності конфігурації. Це дозволяє проводити оперативне внесення змін у конфігурацію безпосередньо через веб-інтерфейс системи (рисунок 3.20).

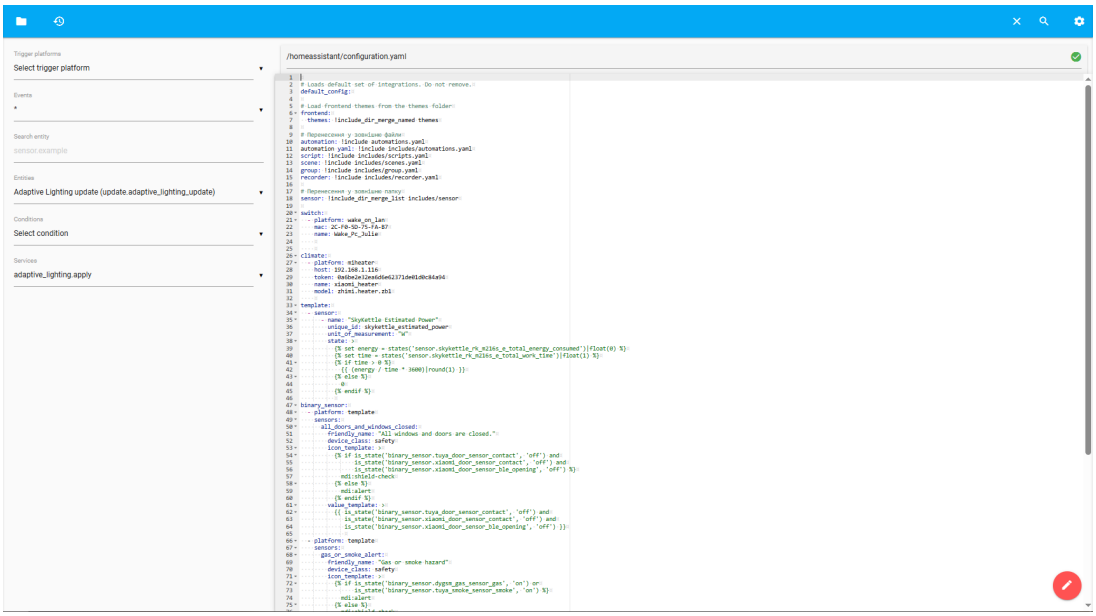

Рис. 3.20: Вбудований файловий редактор із відкритим конфігураційним файлом

Компонент *HACS* виступає спільнотним магазином розширень, який надає доступ до сторонніх інтеграцій, карток інтерфейсу, тем і функціональних модулів. Модуль дозволяє встановлювати, оновлювати та видаляти

розширення безпосередньо через інтерфейс *Home Assistant*. Кожен компонент супроводжується документацією, версіонуванням і рейтингом користувачів, що спрощує вибір і підвищує надійність установлених елементів (рисунок 3.21).

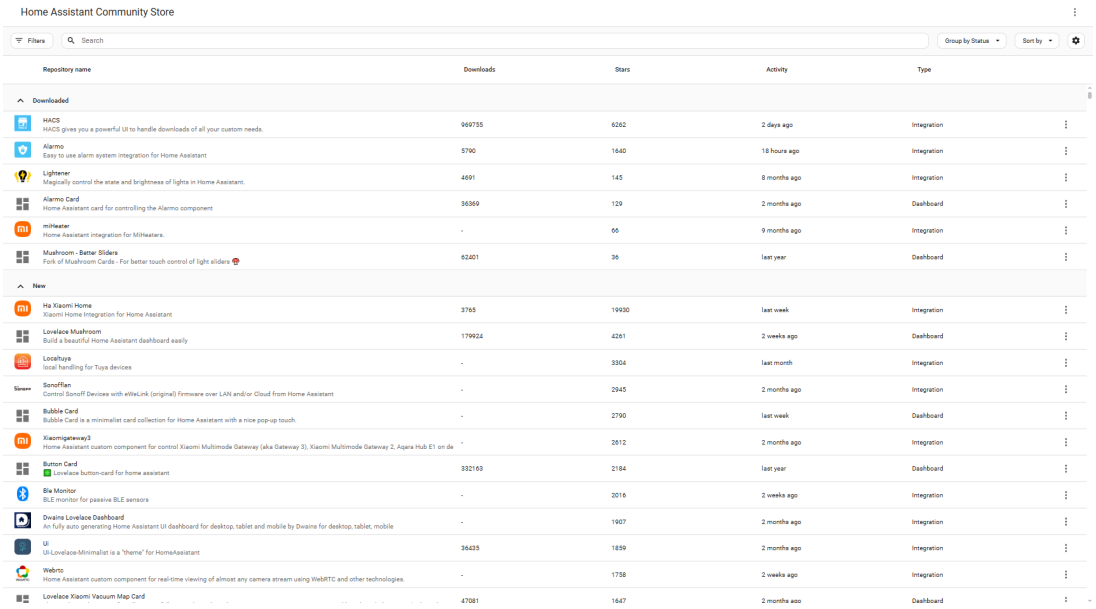

Рис. 3.21: Інтерфейс Home Assistant Community Store (HACS)

Модуль «Terminal & SSH» забезпечує доступ до командної оболонки операційної системи (Linux у віртуалізованому або контейнерному середовищі). Він дозволяє виконувати адміністративні команди, оновлювати компоненти вручну, переглядати журнали подій, керувати службами, діагностувати мережеві підключення та інсталювати додаткові бібліотеки або скрипти. На рисунку 3.22 представлено інтерфейс терміналу.

Рис. 3.22: Модуль «Термінал» із доступом до командної оболонки

Доступ до терміналу захищено багаторівневою автентифікацією, підтримуються SSH-ключі та розмежування прав користувачів. Це забезпечує безпечне адміністрування системи й контроль над усіма процесами технічного обслуговування без необхідності прямого доступу до серверного середовища.

3.1.4. Системні інструменти та персоналізація.

Підсистема системних інструментів і персоналізації забезпечує контроль, адаптацію та налаштування ключових компонентів середовища *Home Assistant*. Її функції охоплюють моніторинг станів сутностей, виклик служб, управління інтеграціями, персональні налаштування користувача та централізовану систему сповіщень. Завдяки цьому забезпечується повна керованість, гнучкість і зручність експлуатації системи автоматизації.

Модуль «Developer Tools» (рисунок 3.23) призначений для моніторингу й ручного керування сутностями (*entities*), виклику служб (*services*), а також тестування логіки автоматизації. Крім того, він надає можливість перевірки синтаксису YAML-файлів, візуалізації поточних станів об'єктів і виконання часткового перезавантаження окремих компонентів системи без необхідності повного рестарту середовища. Це значно спрощує процес розробки, налагодження та тестування автоматизаційних сценаріїв.

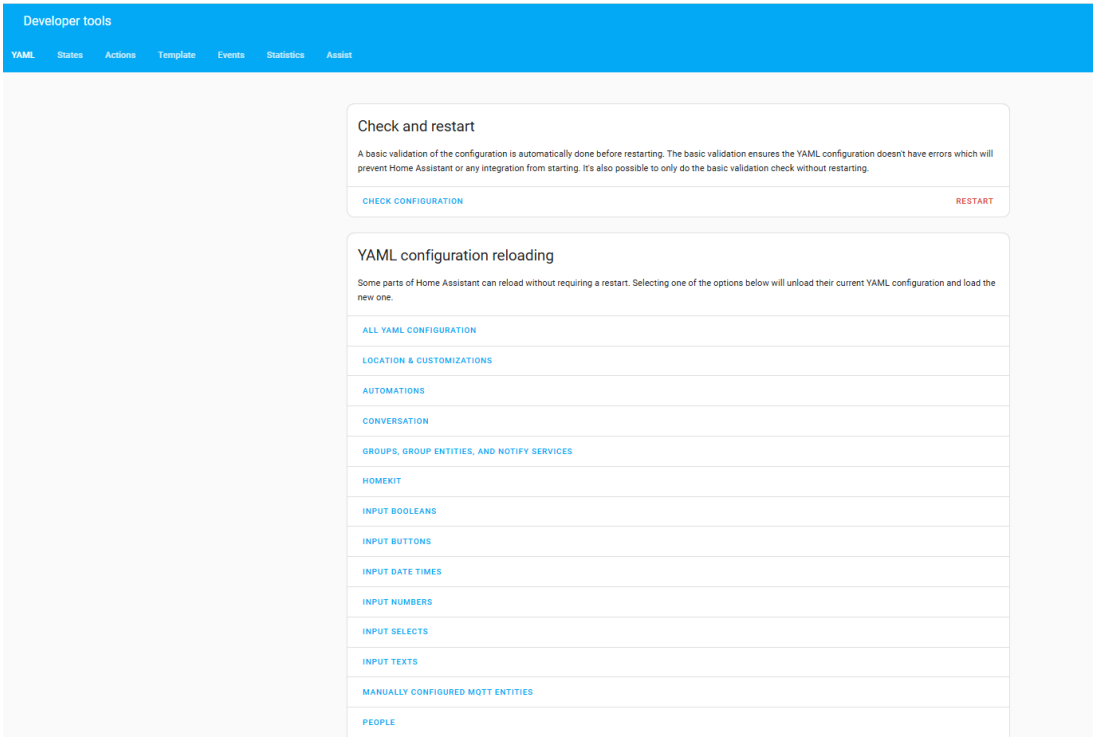

Рис. 3.23: Інструменти розробника: перегляд сутностей і виклик служб

Розділ «Settings» (рисунок 3.24) є центральним адміністративним інтерфейсом для управління параметрами системи. Він охоплює налаштування облікових записів, прав доступу, локалізації, інтеграцій (*MQTT*, *Google Assistant*, *Alexa*, *Philips Hue*, *Zigbee2MQTT*, *Modbus*, *KNX* тощо), а також оновлення ядра й додатків. Крім того, модуль дозволяє конфігурувати мережеві параметри (IP, DHCP, VPN, проксі, SSL/TLS), резервне копіювання, діагностику продуктивності та керування процесами через *Supervisor*. Інтерфейс реалізує логічну ієрархію налаштувань, що підвищує зручність адміністрування.

Модуль «Notifications» (рисунок 3.25) реалізує централізований механізм інформування користувача про події системи, статуси автоматизацій і критичні зміни в роботі компонентів. Підтримуються різноманітні канали доставки — мобільні застосунки, електронна пошта, месенджери, голосові помічники. Журнал сповіщень інтегровано безпосередньо в дашборд, що забезпечує швидкий доступ до історії подій і дозволяє здійснювати аналіз станів системи в реальному часі.

Розділ «User Profile» (рисунок 3.26) містить персональні параметри користувача: ім'я, аватар, пароль, мову інтерфейсу, часовий пояс, формат дати й часу, параметри сповіщень і тему оформлення. Також підтримуються функції багатофакторної автентифікації, відображення активних сесій і підключених пристроїв. Персоналізація інтерфейсу сприяє адаптації середовища до потреб користувача, забезпечуючи зручність і підвищену безпеку при роботі з системою.

3.1.5. Робоча область дашбордів.

Праворуч від бічного меню розташовано основну робочу область інтерфейсу системи, у якій відображаються інформаційні панелі різної функціональної спрямованості (клімат, енергія, безпека, мультимедіа, системні служби тощо). Кожен дашборд може містити набори карток (entities, glance, history-graph, gauge, button, markdown тощо), налаштованих під конкретні сценарії користування. Розміщення карток забезпечує адаптивність до розмірів екрана та ролі користувача, а також підтримує контекстну навігацію між підсистемами.

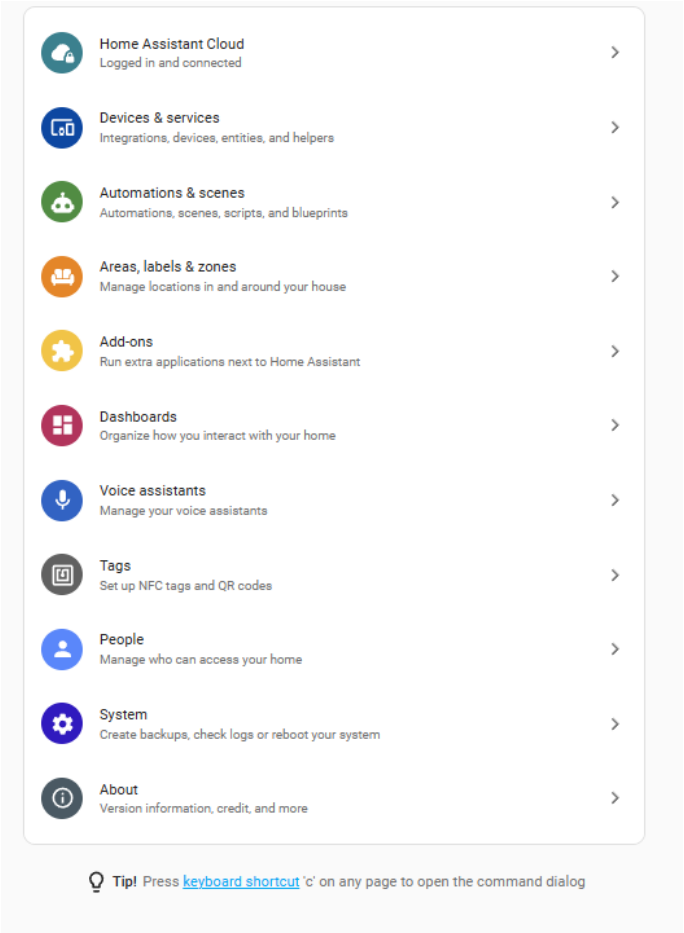

Рис. 3.24: Головне меню налаштувань у Home Assistant

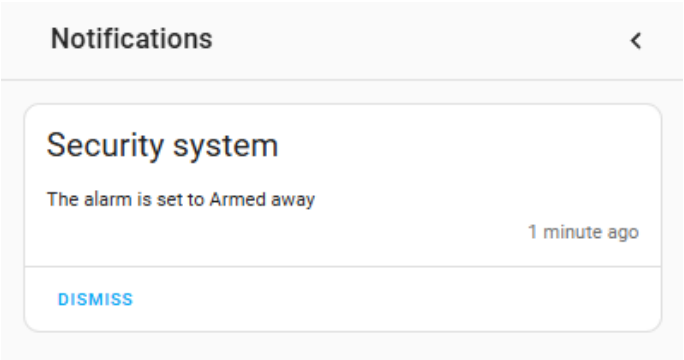

Рис. 3.25: Приклад повідомлення про увімкнення системи безпеки

3.2. Функціональний інтерфейс сторінок підсистем

Праву частину інтерфейсу системи займає основна робоча область, у межах якої розміщено інформаційні панелі з різним функціональним призначенням. Кожна з цих панелей реалізує окрему підсистему моніторингу та керування, утворюючи узгоджений функціональний інтерфейс локальної системи автоматизації.

3.2.1. Інформаційна панель «Climate».

У системі автоматизації середовища на базі *Home Assistant* дашборд «Climate» виконує роль централізованої панелі моніторингу та керування параметрами мікроклімату в приміщеннях (рисунок 3.27). Структура панелі побудована за принципом функціонального розділення інформаційних блоків, що забезпечує одночасну

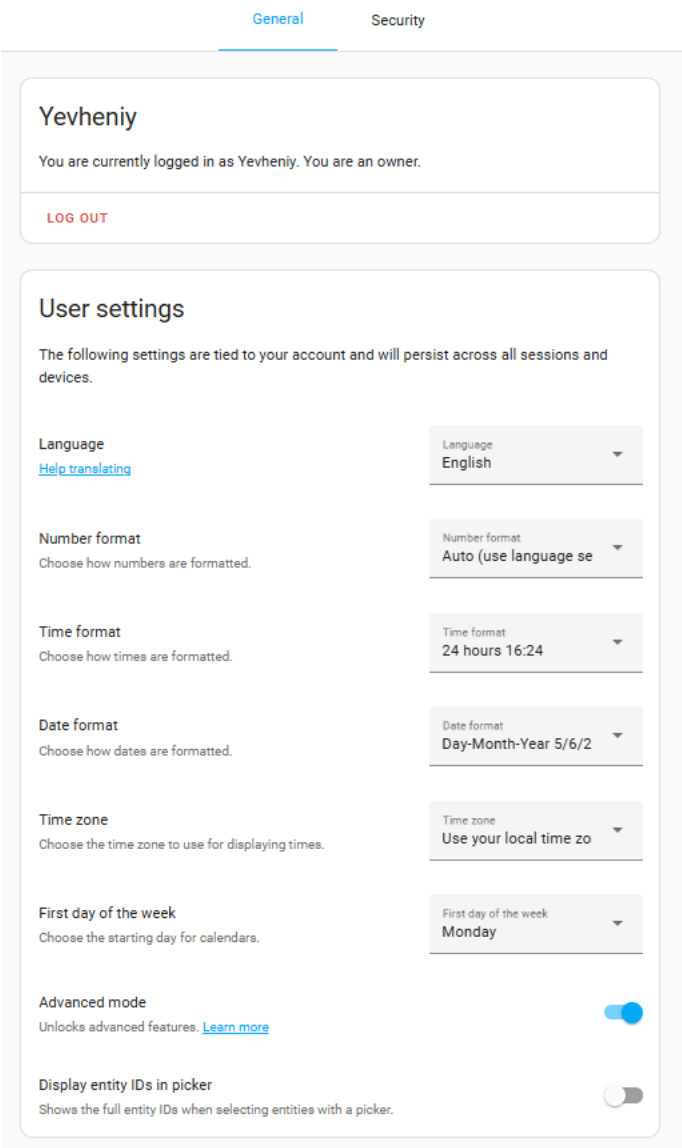

Рис. 3.26: Інтерфейс профілю користувача

візуалізацію зовнішніх і внутрішніх показників, а також інтерактивне керування кліматичними пристроями. Програмний код налаштування інтерфейсу можна знайти у підрозділі Б.1

Зовнішні метеорологічні умови відображаються на основі інтеграції з погодними сервісами, тоді як внутрішні параметри - температура, вологість, тиск, концентрація шкідливих домішок - надходять від мережі локальних сенсорів, підключених через протоколи Zigbee, Wi-Fi та Bluetooth. Такий підхід забезпечує комплексний контроль мікроклімату з високим ступенем точності, просторової деталізації та часової роздільної здатності.

Окрім моніторингу, дашборд реалізує можливості керування ключовими кліматичними пристроями - зволожувачами, вентиляторами, кондиціонерами та обігрівачами - із підтримкою як ручного, так і автоматизованого режимів роботи. Інтерфейс побудований за модульним принципом: окремі функціональні блоки можуть конфігуруватися, додаватися або видалятися відповідно до специфіки застосування системи, зокрема в навчальних, офісних або житлових приміщеннях. Така гнучкість дозволяє адаптувати структуру панелі до потреб користувача та сценаріїв експлуатації. Логіку автоматизації описано в додатку Б.2

У лівій колонці дашборду розміщено блок відображення актуальної метеорологічної інформації, який забезпечує користувача узагальненими даними про зовнішні погодні умови в режимі реального часу (рисунок 3.28). Основними параметрами, що візуалізуються, є температура повітря, відносна вологість, атмосферний тиск, напрямок і швидкість вітру. Усі показники подано у компактній формі із застосуванням піктограм та цифрових

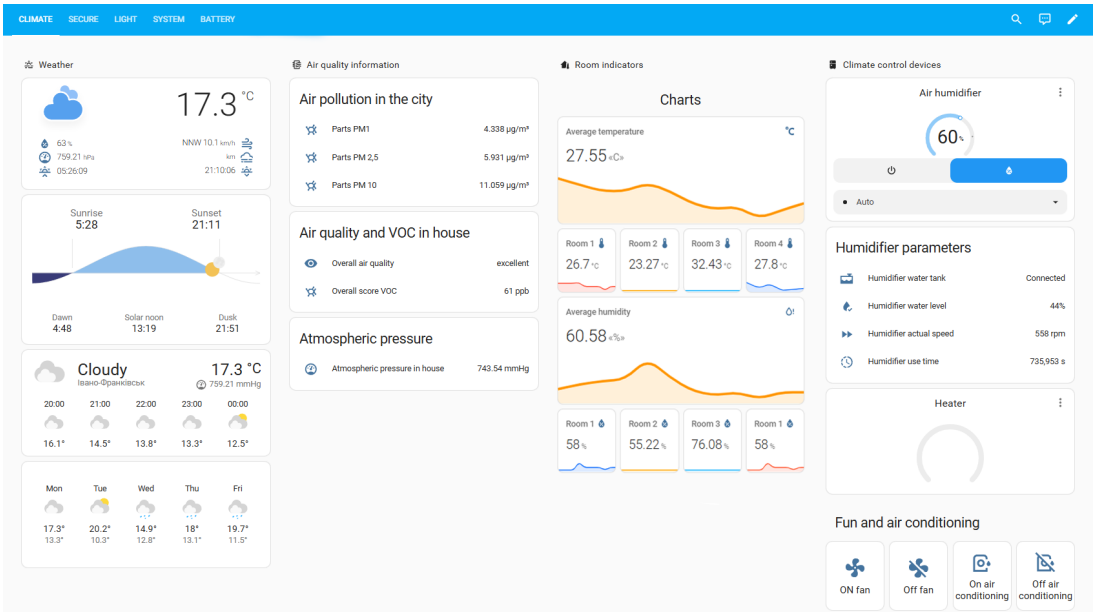

Рис. 3.27: Інтерфейс інформаційної панелі «Climate».

індикаторів, що сприяє швидкому сприйняттю інформації та зручності оперативного аналізу.

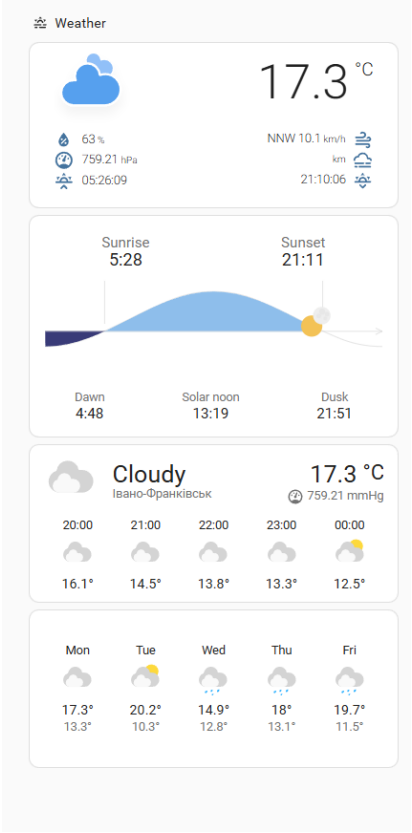

Рис. 3.28: Інтерфейс колонки «Weather» інформаційної панелі «Climate».

Окремий підблок присвячено візуалізації світлового дня. Відповідна діаграма відображає ключові фази добового циклу: час сходу та заходу сонця, момент астрономічного полудня, а також періоди ранкових і вечірніх сутінок. Дані формуються на основі внутрішньої інтеграції *sun*, що входить до базової архітектури *Home Assistant*. Використання цього компоненту забезпечує точний розрахунок астрономічних подій з урахуванням

географічних координат розташування системи, що є критично важливим для коректної роботи світлових та охоронних сценаріїв.

Окрім поточної інформації, блок включає коротко- та довгостроковий прогноз погоди, що охоплює найближчі години та декілька наступних днів. Прогнозні дані надходять через інтеграцію із сервісом *Meteorologisk institutt (Met.no)* - національним метеорологічним агентством Норвегії. Зазначений сервіс характеризується високою достовірністю та деталізацією прогнозних моделей, особливо для європейського регіону, що робить його доцільним джерелом даних для інтелектуальних систем керування мікрокліматом.

З функціональної точки зору, блок «Weather» виконує не лише інформативну, а й керуючу роль, оскільки його показники використовуються як вхідні змінні в контекстно-орієнтованих сценаріях автоматизації. Наприклад, зниження температури нижче заданого порогу може автоматично активувати системи опалення, а перевищення порогових значень швидкості вітру - ініціювати блокування зовнішніх жалюзі чи обмеження роботи зовнішніх зволожувальних систем. Аналогічно, інформація про час заходу сонця може застосовуватись для автоматизованого ввімкнення зовнішнього та внутрішнього освітлення.

У другій колонці дашборду «Climate» розташовано функціональний блок «Air Quality Information», призначений для моніторингу та візуалізації параметрів якості повітря як у зовнішньому, так і у внутрішньому середовищі. Основне призначення цього блоку полягає у формуванні повної інформаційної картини екологічного стану повітря, що створює основу для реалізації адаптивного управління вентиляційними, фільтраційними та іншими кліматичними підсистемами розумного середовища.

Рис. 3.29: Інтерфейс колонки «Air Quality Information» інформаційної панелі «Climate».

Секція *Air pollution in the city* акумулює дані щодо концентрації зважених часток у повітрі (PM1, PM2.5, PM10), які надходять через інтеграційний модуль, пов'язаний із муніципальними або приватними системами моніторингу якості повітря. Інформація відображається в режимі реального часу, що дає змогу аналізувати динаміку рівня забруднення, фіксувати перевищення гранично допустимих концентрацій та ухвалювати рішення, спрямовані на зменшення впливу зовнішніх чинників на внутрішнє середовище. Вказані параметри можуть застосовуватись у сценаріях автоматизації - зокрема, для обмеження припливу зовнішнього повітря, активації систем очищення або зміни режимів вентиляції при перевищенні критичних значень.

Внутрішні екологічні параметри представлені у секції *Air quality and VOC in house*, де візуалізується узагальнений індекс якості повітря (AQI) та рівень летких органічних сполук (VOC). Ці показники характеризують стан повітряного середовища всередині приміщень та відображають його зміну під впливом побутових процесів, режимів вентиляції, інтенсивності перебування людей та використання хімічних засобів. Аналіз цих даних

дозволяє своєчасно виявляти погіршення мікроклімату та реалізовувати адаптивне керування зволоженням, провітрюванням та очищенням повітря.

Окремий модуль *Atmospheric pressure* забезпечує відображення поточного значення атмосферного тиску, який виступає додатковим параметром для оцінки мікроклімату. У поєднанні з іншими показниками він може використовуватися для побудови моделей комфортності середовища та оцінки потенційного впливу метеорологічних факторів на самопочуття користувачів.

Таким чином, блок «Air Quality Information» формує цілісну систему моніторингу повітряного середовища у зовнішньому та внутрішньому контекстах. Його інтеграція до дашборду «Climate» забезпечує доступ до актуальної, багаторівнево структурованої інформації та підтримує прийняття контекстно-залежних рішень у системі локальної автоматизації. Це сприяє підвищенню екологічної безпеки, покращенню якості життя та оптимізації енергоефективності приміщень.

Функціональний блок «Room Indicators» у третій колонці дашборду «Climate» виконує роль комплексного аналітичного модуля для моніторингу мікрокліматичних параметрів у приміщеннях (рисунок 3.30). Його основне призначення полягає у візуалізації, аналізі та інтерпретації даних щодо температури, вологості й тиску, які в сукупності визначають загальний стан внутрішнього середовища будівлі. Завдяки структурованій подачі інформації користувач має змогу оперативно оцінювати поточні умови, ідентифікувати відхилення від рекомендованих значень та реагувати на них за допомогою автоматизованих сценаріїв.

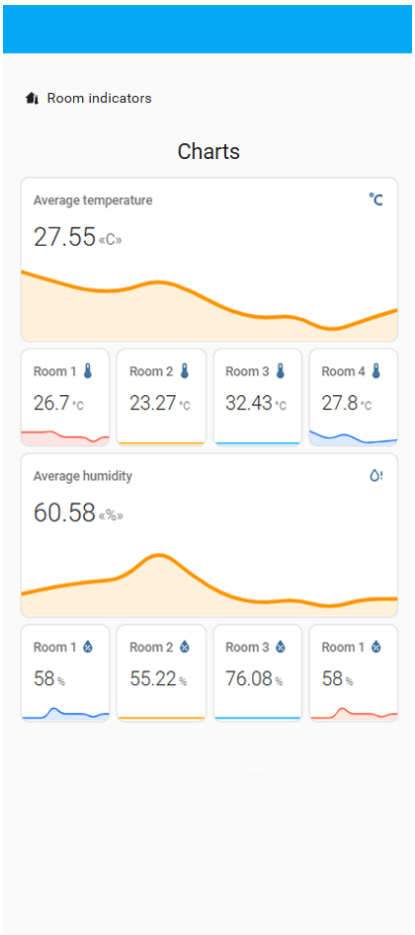

Рис. 3.30: Інтерфейс колонки «Room Indicators» інформаційної панелі «Climate».

Інтерфейс «Room Indicators» має дворівневу структуру. На першому рівні відображаються агреговані середні значення температури та вологості по будівлі загалом, що дає змогу сформувати узагальнену оцінку мікроклімату. Другий рівень забезпечує деталізацію по окремих зонах - житлових, технічних і допоміжних приміщеннях - де дані подаються у вигляді динамічних графіків та індикаторів. Така багаторівнева візуалізація

поєднує оглядову аналітику з можливістю локалізації проблемних ділянок, які можуть свідчити про неефективність вентиляції, порушення теплообміну або некоректну роботу обладнання.

Дані для цього модуля надходять із локальної сенсорної мережі, що функціонує на основі енергоефективних бездротових протоколів. Архітектура мережі забезпечує просторове покриття всіх функціональних зон будівлі та дає змогу здійснювати безперервний збір даних у режимі реального часу. Паралельно система виконує діагностичні функції, аналізуючи стабільність каналів зв'язку, частоту оновлення та достовірність отриманих значень. Це створює можливості для оцінки надійності інфраструктури, прогнозування навантажень та оптимізації енергоспоживання.

Особливе значення мають алгоритми аналітичної обробки даних. На основі накопичених історичних записів модуль формує тренди, що дозволяють виявляти сезонні та добові закономірності зміни мікрокліматичних показників, а також порівнювати параметри для різних приміщень. У разі фіксації критичних змін (надмірне зниження температури, підвищення вологості тощо) система може автоматично активувати відповідні сценарії, зокрема, регулювання потужності опалення, режимів вентиляції або надсилання сповіщень користувачам.

Отже, блок «Room Indicators» виступає ключовим елементом інформаційно-аналітичної підсистеми локальної автоматизації. Він забезпечує не лише відображення поточних умов, а й створює основу для інтелектуального прийняття рішень у межах розумного середовища. Інтеграція цього модуля до панелі «Climate» підвищує ефективність управління мікрокліматом, сприяє підтриманню стабільних екологічних параметрів та формуванню комфортного й енергоефективного простору.

У четвертій колонці дашборду «Climate» розташовано блок «Climate Control Devices», який виконує роль інтерактивного центру керування кліматичними пристроями, інтегрованими до системи через різні шлюзи, протоколи зв'язку та зовнішні API (рисунок 3.31). Основна функція цього блоку полягає не лише у наданні користувачеві зручного інтерфейсу для ручного контролю, а й у забезпеченні сценарної взаємодії пристроїв у межах автоматизацій *Home Assistant*.

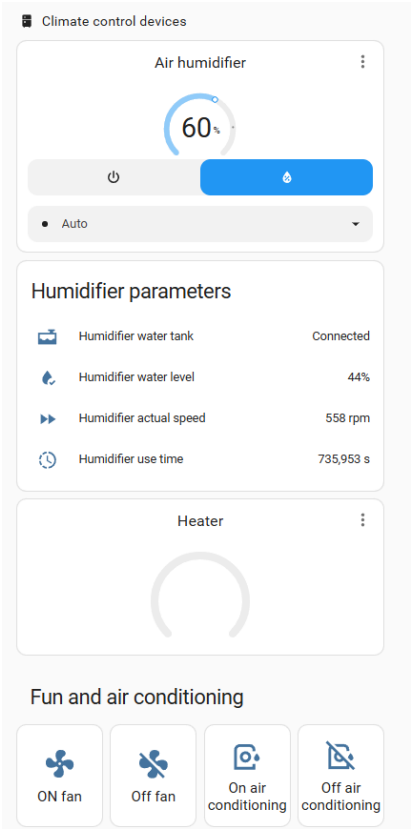

Рис. 3.31: Інтерфейс колонки «Climate Control Devices» інформаційної панелі «Climate».

У межах цього блоку представлено три основні групи пристроїв: зволожувачі повітря, обігрівачі та засоби активного охолодження (вентилятори, кондиціонери). Для кожного елемента передбачено індивідуальний візуальний інтерфейс, який відображає поточний стан пристрою, дозволяє змінювати режими роботи, вмикати або вимикати обладнання в ручному режимі, а також включати його до сценаріїв автоматизації з урахуванням поточних параметрів середовища.

Керування зволожувачем повітря реалізовано за допомогою інтеграції *Xiaomi Miio*, яка надає доступ до основних функціональних параметрів: встановлення цільового рівня вологості, вибору інтенсивності роботи, моніторингу рівня заповнення резервуара та перемикання між автоматичним і ручним режимами. Наявність зворотного зв'язку дозволяє системі своєчасно реагувати на критичні стани, такі як відсутність води, та адаптувати режими роботи пристрою до поточних внутрішніх і зовнішніх умов.

Керування обігрівачем реалізовано через інтеграцію *miHeater*, що забезпечує параметричне керування режимами роботи пристрою. Користувач може задавати бажану температуру в приміщенні, встановлювати таймери, контролювати активний режим (економний, інтенсивний, режим очікування), а також відстежувати орієнтовні показники споживання енергії. Інтеграція підтримує автоматизоване вимкнення при досягненні цільового мікроклімату, що є важливим з точки зору енергоефективності та безпеки експлуатації.

Вентилятори та кондиціонери інтегруються до системи за допомогою універсального Wi-Fi–інфрачервоного пульта *Broadlink RM Pro*. Цей пристрій виконує емуляцію стандартних ІЧ-команд побутових приладів, які не мають власного мережевого інтерфейсу. Інтеграція *Broadlink* з *Home Assistant* дозволяє формувати набір віртуальних кнопок, кожна з яких відповідає конкретній команді фізичного пульта (наприклад, «увімкнути охолодження», «збільшити швидкість обдуву», «змінити режим роботи»). У результаті система набуває можливості повноцінного керування навіть застарілими або недорогими моделями кліматичної техніки.

Важливою особливістю блоку «Climate Control Devices» є підтримка гібридної моделі керування, що поєднує автоматизовані сценарії та ручне втручання користувача. У разі потреби користувач може змінити стан пристрою або його параметри, не порушуючи глобальної логіки роботи системи. Наприклад, при зниженні вологості нижче 35 % система автоматично активує зволожувач, але водночас користувач має можливість змінити режим роботи пристрою або тимчасово вимкнути його.

Інтерфейс керування кліматичними приладами підтримує зворотний моніторинг стану: на дашборді відображаються не лише індикатори ввімкнення, а й поточні значення основних параметрів (температура, вологість, режим роботи), представлені у вигляді інтерактивних віджетів. Це створює умови для адаптивного реагування системи, підвищує прозорість функціонування та сприяє безпечній експлуатації обладнання.

Таким чином, блок «Climate Control Devices» є ключовим компонентом у структурі системи локального кліматичного контролю. Він поєднує зручність керування, багатопротокольну інтеграцію, розширену автоматизацію та підтримку широкого спектра побутових і напівпрофесійних кліматичних пристроїв у межах єдиного інтерфейсу *Home Assistant*.

3.2.2. Інформаційна панель «Security».

У системі автоматизації середовища на базі *Home Assistant* дашборд «Security» виконує роль інтегрованого центру контролю безпеки об'єкта. Його структура реалізована за принципом модульного поділу функціональності, що охоплює ключові аспекти фізичної та техногенної безпеки: охоронні системи, сенсори витоку газу та води, виявлення руху, відеоспостереження, а також контроль доступу через вікна й двері. Програмний код налаштування інтерфейсу можна знайти у підрозділі В.1.

Формування даних у дашборді базується на поєднанні локальних сенсорів (підключених через Zigbee, Wi-Fi або Bluetooth) і зовнішніх джерел інформації, зокрема державних та міських систем сповіщення про тривоги. Такий підхід дає змогу формувати багаторівневу модель безпеки, що охоплює як внутрішні приміщення, так і зовнішній периметр об'єкта.

Інтерфейс побудовано з урахуванням вимог до швидкої реакції, точного локалізованого моніторингу та автоматизованого сценарного реагування. Кожен елемент дашборду дозволяє не лише відстежувати стан об'єктів або зон, а й втручатися в роботу системи в режимі реального часу - наприклад, активувати сигналізацію, переглянути відеопотік з камери чи проаналізувати історію руху у контрольованих областях (рисунок 3.32).

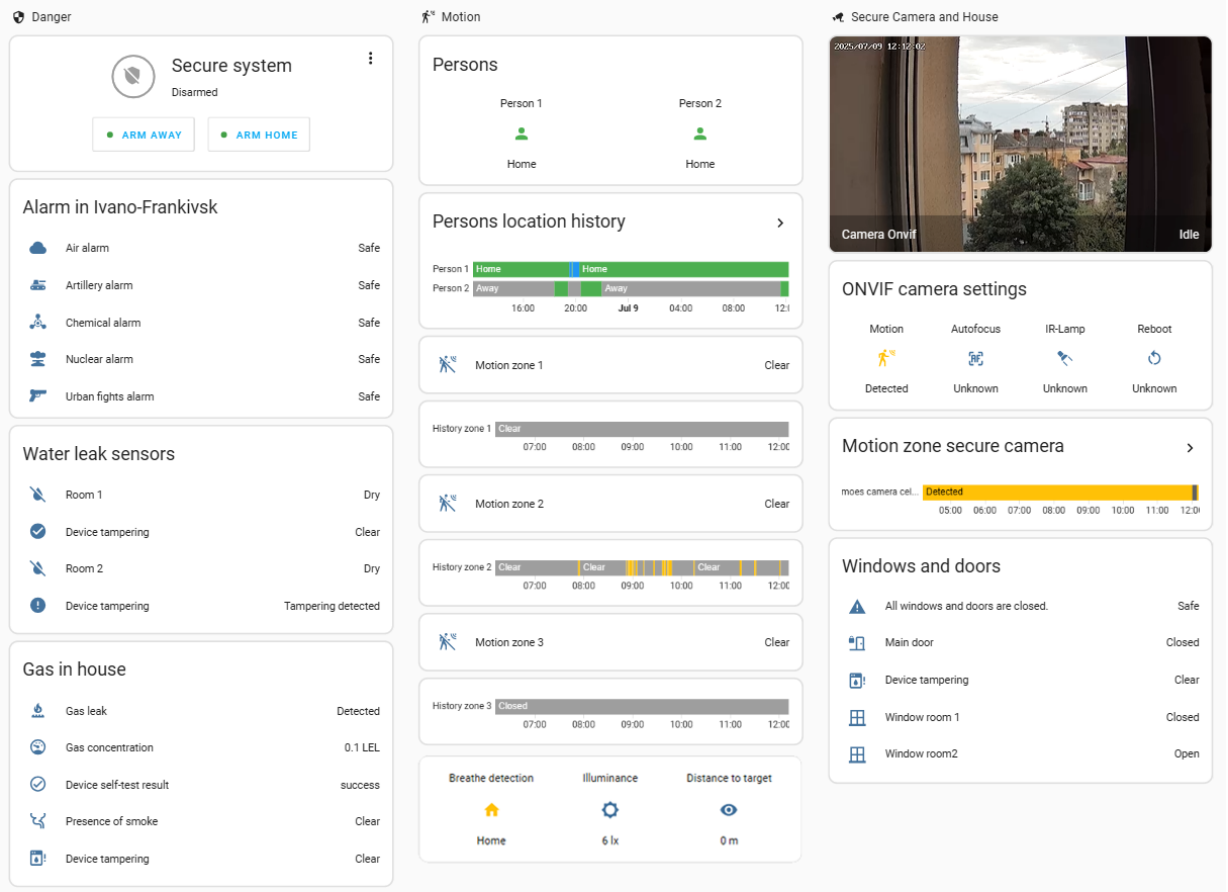

Рис. 3.32: Інтерфейс інформаційної панелі «Security».

Завдяки гнучкій інтеграції та підтримці різних протоколів зв'язку, дашборд «Security» може адаптуватися до потреб житлових, комерційних та навчальних об'єктів, забезпечуючи високий рівень безпеки, прозорості та централізованого управління в межах єдиної цифрової платформи. Логіку автоматизації описано в додатку В.2.

Перша колонка дашборду «Security» виконує функцію критично важливого інформаційно-керувального блоку, що акумулює засоби запобігання, виявлення та реагування на фізичні й техногенні загрози в автоматизованому середовищі. Її структура охоплює як внутрішні охоронні засоби, так і зовнішні інформаційні джерела, забезпечуючи системну багаторівневу безпеку (рисунок 3.33).

Центральним компонентом функціонального блоку *Danger* є панель *Secure System*, що реалізує керування повнофункціональною охоронною сигналізацією на базі інтеграції *Alarmo*. Застосування цього модуля забезпечує широкі можливості щодо конфігурації сценаріїв охорони, зонального поділу об'єкта, а також розмежування прав доступу між користувачами.

Панель підтримує три базові стани сигналізації:

- *Arm Away* - повний режим охорони, що активується у випадку повної відсутності людей у приміщенні та передбачає задіяння всіх наявних сенсорів;
- частковий режим охорони, за якого окремі сенсори (наприклад, руху у житлових зонах) можуть бути тимчасово ігноровані, зберігаючи контроль за периметром;
- режим деактивації сигналізації, за якого всі автоматизації та тривожні сповіщення призупинено.

Інтеграція *Alarmo* дозволяє задавати часові затримки на вхід/вихід, формувати власні охоронні зони, що включають або виключають окремі групи сенсорів (наприклад, лише двері або лише датчики руху), а також керувати доступом через персоналізовані PIN-коди для різних користувачів. Це забезпечує високий ступінь гнучкості в налаштуванні охоронної логіки.

У межах автоматизацій *Home Assistant* підсистема *Secure System* може взаємодіяти з іншими пристроями,

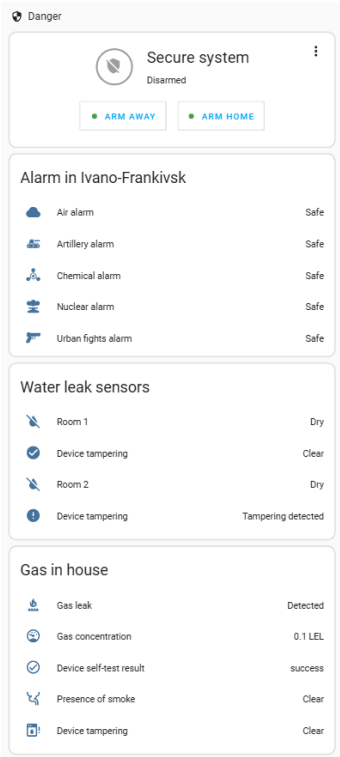

Рис. 3.33: Інтерфейс колонки «Danger» інформаційної панелі «Security».

виконуючи роль центрального тригера сценаріїв реагування. Наприклад, перехід у режим *Arm Away* може викликати:

- увімкнення зовнішнього відеоспостереження з одночасною активацією запису;
- вимкнення внутрішнього освітлення в неактивних зонах;
- знеструмлення другорядних електроприладів;
- активацію сирени при спрацюванні будь-якого сенсора.

Блок *Alarm in Ivano-Frankivsk* розширює охоронну логіку за межі приватного простору, реалізуючи моніторинг загальнодержавних та регіональних загроз, які можуть впливати на поведінку автоматизованої системи. Джерелом інформації є інтеграція *Ukraine Alarm*, що отримує актуальні дані з офіційних або волонтерських джерел, синхронізованих із системою повітряних тривог. Система обробляє п'ять типів загроз:

- *Air Alarm* - повітряна тривога (ракетна або авіаційна загроза);
- *Artillery Alarm* - загроза обстрілу з важкої артилерії;
- *Chemical Alarm* - потенційне застосування хімічної зброї;
- *Nuclear Alarm* - загроза або факт ядерної атаки чи аварії;
- *Urban Fights Alarm* - інформація про бойові дії в межах міста.

Користувач має можливість динамічно змінювати геолокацію об'єкта моніторингу - як вручну, так і за допомогою автоматичних сервісів геолокації *Home Assistant*. Це дає змогу застосовувати єдиний шаблон автоматизацій для різних об'єктів з урахуванням їхнього фактичного місцезнаходження. Інформація з блоку *Alarm in Ivano-Frankivsk* може бути інтегрована у складні сценарії реагування, наприклад:

- автоматичне ввімкнення режиму повної охорони під час оголошення повітряної тривоги;
- звукове та візуальне сповіщення про небезпеку;
- активація попереджувального освітлення приміщення при виявленні загроз у конкретній місцевості;
- відключення некритичних систем задля економії енергії та зниження ризиків.

Таким чином, відповідний блок виступає не лише візуалізатором зовнішніх ризиків, а й елементом інтегрованого ланцюга безпеки, що дозволяє адаптувати внутрішню логіку роботи системи до змін геополітичного чи техногенного контексту. Його застосування є особливо актуальним у навчальних закладах, адміністративних

установах та на об'єктах критичної інфраструктури, де необхідні своєчасне інформування та автономна реакція на зовнішні загрози.

Окремим функціональним блоком дашборду є *Water Leak Sensors*, який відповідає за виявлення витоків води у критичних зонах об'єкта моніторингу. Основне завдання цього блоку полягає у своєчасному виявленні аварійних ситуацій, пов'язаних із протіканням водопровідних, каналізаційних або опалювальних систем, із подальшим запуском сценаріїв реагування - від сповіщення користувачів до відключення живлення або активації електроклапанів для перекриття подачі води. Технічна реалізація блоку базується на використанні мережі бездротових сенсорів, інтегрованих через протокол Zigbee, що забезпечує низьке енергоспоживання, стійкість до перешкод і можливість автономної роботи у випадку тимчасової втрати зв'язку з центральним сервером. Серед додаткових функцій сенсорів варто відзначити наявність тамперного контролю, який дозволяє фіксувати спроби фізичного втручання: зняття або переміщення пристрою, відкриття корпусу чи несанкціоновану зміну положення. Це підвищує рівень контролю над системою та забезпечує захист від несанкціонованого доступу.

З огляду на специфіку зон моніторингу, зокрема технічних, підвальних або важкодоступних приміщень, доцільним є використання саме Zigbee-сенсорів замість Bluetooth-рішень, оскільки останні потребують наявності BLE-шлюзів у межах стабільної зони покриття, що не завжди можливо реалізувати. Таким чином, блок *Water Leak Sensors* виконує функцію превентивного контролю техногенних ризиків, пов'язаних із водними витоками, які потенційно можуть призвести до ушкодження інфраструктури або значних матеріальних збитків. Наявність системи тамперингу формує додатковий рівень захисту та забезпечує моніторинг не лише аварій, а й потенційних спроб саботажу або фізичного втручання.

У нижній частині колонки дашборду розміщено блок *Gas in House*, призначений для моніторингу рівня загазованості середовища. Його функціональне завдання полягає у виявленні небезпечних концентрацій горючих або токсичних газів, що можуть створювати ризики займання, вибуху чи отруєння. Блок реалізує комбінований контроль газового середовища, зокрема визначення концентрації природного газу та чадного газу (CO). Сенсори оснащено вбудованими сиренами для локального звукового сповіщення про перевищення допустимих порогів концентрації. Завдяки цьому забезпечується автономність функціонування навіть у випадках тимчасової втрати мережевого з'єднання або джерела живлення. Додатково реалізовано функцію самодіагностики, яка контролює працездатність сенсорів і сповіщає про технічні збої або необхідність обслуговування.

Завдяки інтеграції з *Home Assistant* блок *Gas in House* підтримує формування сценаріїв автоматичного реагування, таких як активація систем примусової вентиляції, відключення живлення або надсилання екстрених сповіщень. Це дозволяє вибудувати багаторівневу систему безпеки, що діє як на рівні локального сенсорного моніторингу, так і в межах глобальної автоматизаційної логіки. Таким чином, *Gas in House* формує додатковий рівень екологічної та техногенної безпеки, особливо важливий для замкнених або слабо вентильованих приміщень. Його застосування є доцільним у котельнях, лабораторіях, кухнях, технічних і житлових просторах, де критично важливим є своєчасне виявлення газових витоків і забезпечення автономного оповіщення.

Друга колонка дашборду «Security» виконує роль підсистеми моніторингу присутності мешканців та фіксації руху в контрольованих зонах (рисунок 3.34). Її функціональна структура включає персоналізовану ідентифікацію осіб, історію їх перебування, а також візуалізацію даних із датчиків руху, розташованих у різних кімнатах або зонах доступу. Такий підхід забезпечує не лише підвищення рівня безпеки, а й формування основи для адаптивних автоматизацій, наприклад, на основі логіки «якщо вдома нікого немає - активувати охорону». Перший функціональний блок колонки *Motion* забезпечує контекстно-залежне виявлення присутності осіб у контрольованому середовищі. У поточній реалізації дашборду відображено статуси присутності двох користувачів (Person 1, Person 2), що дає змогу швидко оцінити, хто перебуває вдома, а хто - відсутній. Цей статус формується на основі даних, що надходять із різних типів сенсорів присутності та геолокаційних інтеграцій, пов'язаних із персональними пристроями.

До основних способів визначення присутності належать:

- підключення мобільного телефону або ноутбука до домашньої Wi-Fi-мережі - один із найпоширеніших тригерів, що не потребує додаткового обладнання; виявлення MAC-адреси в мережі дозволяє системі *Home Assistant* з високою ймовірністю ідентифікувати присутність користувача;

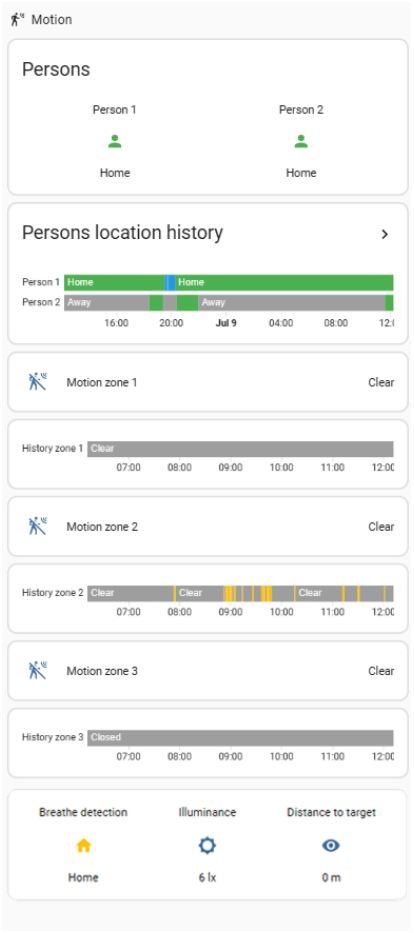

Рис. 3.34: Інтерфейс колонки «Motion» інформаційної панелі «Security».

- використання Bluetooth-маяків і BLE-сканування для виявлення персональних пристроїв, таких як смарт-годинники, бездротові навушники або фітнес-трекери; завдяки цьому навіть за відсутності активного Wi-Fi-з'єднання можлива ідентифікація фізичної присутності людини в зоні дії;
- геолокаційне позиціонування через *Companion App* або GPS-трекери, що дозволяє відстежувати не лише факт перебування вдома, а й місцеперебування в ширшому контексті. Це реалізується шляхом створення віртуальних зон (zones), таких як «дім», «робота», «магазин», «спортзал» тощо.

У випадку активованої геозональної логіки система може реагувати на зміну місця перебування користувача. Наприклад, якщо Person 1 залишає зону «робота», система може автоматично:

- надіслати push-сповіщення іншим членам домогосподарства (наприклад: «Person 1 виїхав з роботи»);
- увімкнути кондиціонер або опалення за певний час до прогнозованого повернення;
- змінити режим освітлення чи активувати охоронну систему залежно від часу доби та інших контекстних факторів.

Блок виявлення присутності виконує не лише роль індикатора поточного статусу користувачів, а й функцію активного компонента системи автоматиза

3.2.3. Інформаційна панель «Light».

Інформаційна панель «Light» у системі *Home Assistant* призначена для централізованого моніторингу, аналізу та керування системою освітлення в приміщеннях. Вона поєднує дані з сенсорів освітленості, станів світильників та виконавчих пристроїв, забезпечуючи користувачеві повний контроль над штучним і природним освітленням у межах локального середовища автоматизації (рисунок 3.35).

Панель реалізує концепцію інтегрованого керування освітленням, у межах якої дані з фізичних сенсорів та астрономічних параметрів поєднуються з механізмами інтелектуальної автоматизації. Це дає змогу адаптувати

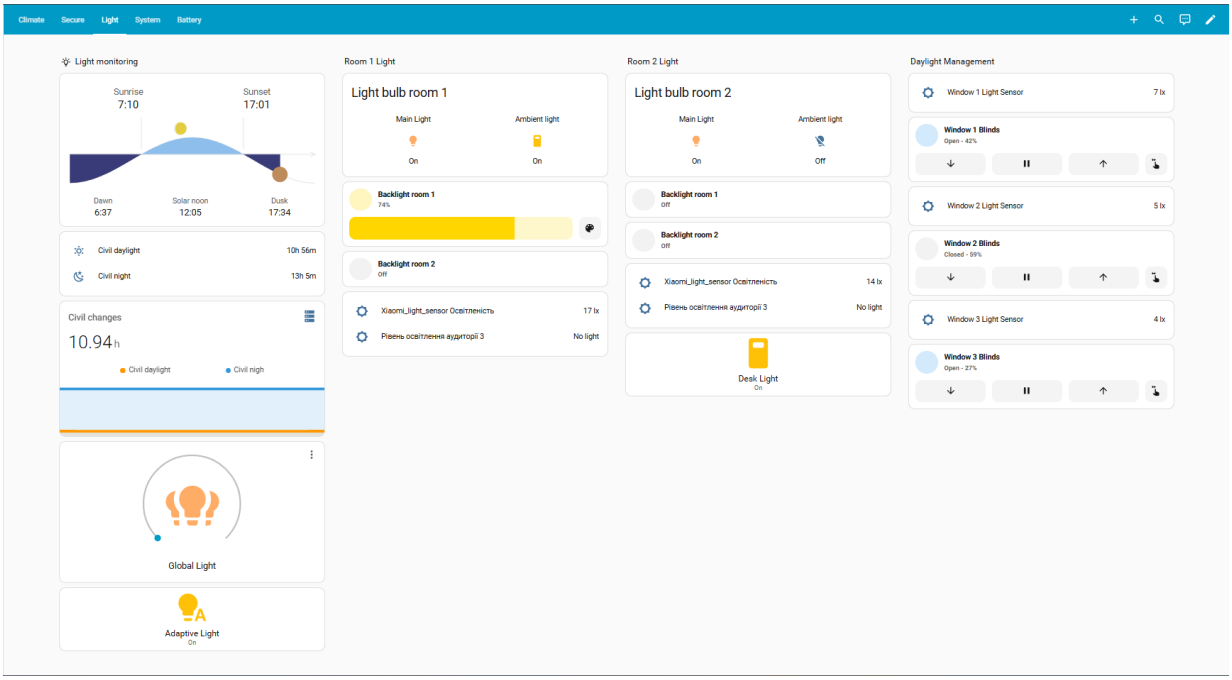

Рис. 3.35: Інтерфейс інформаційної панелі «Light».

роботу освітлювальних пристроїв відповідно до часу доби, рівня природного світла та сценаріїв використання приміщень. Інтерфейс побудований за принципом модульної структури, що забезпечує наочне групування інформації за зонами або приміщеннями. Користувач має можливість відстежувати поточний стан освітлення, обирати автоматичні чи ручні режими керування та взаємодіяти із системою в режимі реального часу через інтерактивні елементи керування.

Таким чином, панель «Light» виконує не лише візуалізаційну, а й ключову керувальну функцію, оскільки слугує основним інструментом для формування та тестування логіки автоматизацій, спрямованих на підвищення енергоефективності, ергономічності освітлення й комфортності середовища для користувачів. Вона інтегрує сенсорні дані, керування пристроями та алгоритми адаптації в єдиному інтерфейсі, забезпечуючи комплексне управління світловим середовищем навчальних і робочих просторів.

Колонка *Light Monitoring* реалізує функціональний модуль моніторингу природного та штучного освітлення в межах автоматизованої системи. Її призначення полягає в забезпеченні аналітичного контролю за світловими умовами середовища, адаптації сценаріїв освітлення до добових циклів та підтриманні оптимального рівня освітленості для енергоефективної роботи системи. Структурно колонка поєднує три логічні блоки: астрономічний календар добового циклу, зведену аналітику світлових фаз і інтегроване керування освітленням на основі адаптивної логіки.

Перший інформаційний блок колонки відображає астрономічні параметри добового циклу (рисунок 3.36), зокрема часи світанку, сонячного полудня, заходу сонця, тривалість світлового дня та нічного періоду. Зазначені дані формуються на основі географічних координат, заданих у системі *Home Assistant*, і автоматично оновлюються відповідно до поточної дати. Візуалізація реалізована у вигляді динамічного графіка із позначенням фаз *Dawn*, *Solar Noon*, *Dusk*, що дає змогу користувачеві інтуїтивно оцінювати тривалість і сезонні зміщення світлових періодів протягом року.

Другий блок узагальнює статистику тривалості світлового та нічного періодів у годинах, що дає змогу відстежувати сезонну динаміку зміни освітленості. На основі цих параметрів система може автоматично адаптувати сценарії освітлення - наприклад, подовжувати період роботи вуличних або фасадних світильників у зимовий час, коли природне освітлення скорочується. Таким чином формується основа для інтелектуальної взаємодії між екологічними параметрами середовища й енергоспоживанням.

Третій функціональний блок забезпечує безпосереднє керування штучним освітленням. Відповідний ін-

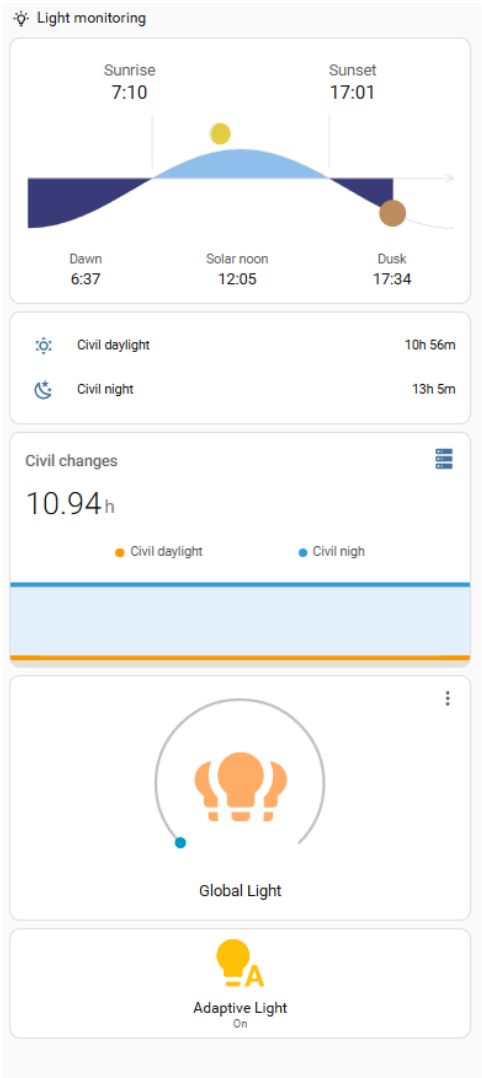

Рис. 3.36: Інтерфейс колонки «Light Monitoring» із відображенням фаз добового циклу.

терфейсний елемент відображає поточний стан головного освітлення в системі та дозволяє здійснювати його активацію або деактивацію як у ручному режимі, так і через автоматизовані правила.

Елемент *Adaptive lighting* відповідає за динамічне регулювання яскравості та колірної температури світильників у реальному часі залежно від фази добового циклу. Ця функція базується на алгоритмах циркадного освітлення, що сприяють підтриманню природних біоритмів користувачів, підвищенню рівня концентрації вдень і створенню комфортних умов відпочинку у вечірній час. Адаптивне керування реалізується через інтеграцію з компонентом *Adaptive Lighting Home Assistant*, який формує профілі освітлення за параметрами часу, геолокації та поточної освітленості середовища. Система може враховувати рівень природного освітлення, визначений фотометричними сенсорами або погодними сервісами, що дозволяє мінімізувати надмірне споживання електроенергії без втрати комфортності умов. Завдяки поєднанню візуальної аналітики, астрономічних розрахунків і адаптивних механізмів керування модуль *Light Monitoring* виконує подвійну функцію - інформаційно-аналітичну та регуляторну. Його застосування дає змогу підвищити енергоефективність системи, забезпечити екологічну сталість і створити інтелектуально адаптивне освітлювальне середовище, що узгоджується з принципами концепції *smart environment*.

Колонка *Room 1 Light* виконує функцію локального моніторингу й керування системою освітлення в межах конкретного приміщення (рисунок 3.37). Вона поєднує сенсорну аналітику, засоби регулювання інтенсивності світла та параметри керування кольоровою підсвіткою, забезпечуючи повний цикл контролю освітлення - від

вимірювання до адаптивної реакції системи в реальному часі.

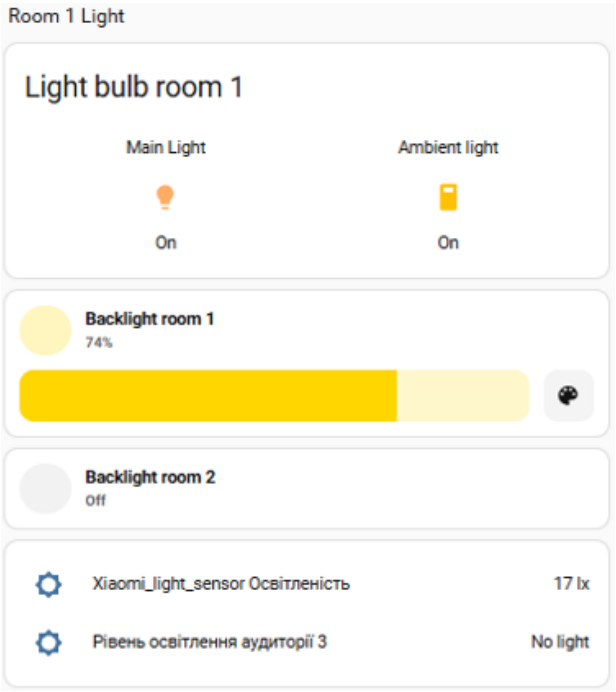

Рис. 3.37: Інтерфейс колонки «Room 1 Light».

Перший блок відображає основні елементи керування для головного (*Main Light*) та допоміжного (*Ambient Light*) освітлення, демонструючи їхній поточний стан за допомогою інтуїтивних піктограм. Користувач може безпосередньо змінювати статус кожного джерела світла або інтегрувати їх у сценарії автоматизації, наприклад для синхронного вмикання під час настання сутінків або зниження рівня природного освітлення.

Другий блок *Backlight room 1* забезпечує регулювання інтенсивності допоміжного освітлення з точністю до одного відсотка. Динамічна шкала відображає поточний рівень яскравості (у наведеному прикладі - 74 %), що дає змогу гнучко налаштовувати світловий комфорт у приміщенні. Додатково реалізовано можливість зміни кольору підсвітки, завдяки чому користувач може формувати різні сценарії освітлення - від нейтрального робочого білого до теплих або декоративних відтінків. Поруч розташовано елемент *Backlight room 2*, який у поточному стані вимкнений, проте може бути задіяний у групових автоматизаціях для симетричного або зонального керування освітленням.

У нижній частині колонки розміщено аналітичні віджети сенсорів освітленості, інтегрованих через шлюз *Zigbee2MQTT*. Зокрема, сенсор *Xiaomi light sensor* фіксує поточний рівень освітленості в люксах, тоді як локальний індикатор *Рівень освітлення аудиторії 3* надає узагальнену якісну оцінку стану освітлення (у наведеному прикладі - «No light»). Завдяки цим показникам система здійснює моніторинг не лише факту наявності світла, а й його інтенсивності, що є основою для контекстно-залежних автоматизацій. Наприклад, при падінні освітленості нижче заданого порогу система автоматично збільшує яскравість або активує додаткову підсвітку.

Взаємодія між сенсорами, виконавчими елементами та інтерфейсом користувача здійснюється в межах локальної мережі без залучення хмарних сервісів, що гарантує мінімальні затримки реакції й підвищену надійність роботи системи. Передбачено двосторонню синхронізацію станів: будь-яка зміна, виконана вручну або через автоматизацію, миттєво відображається в інтерфейсі *Lovelace UI*.

Отже, колонка *Room 1 Light* є комплексним модулем інтелектуального освітлення, який поєднує моніторинг, аналітику, регулювання яскравості та керування кольором світла. Її реалізація сприяє підвищенню енергоефективності, забезпеченню ергономічного комфорту та створює передумови для впровадження персоналізованих сценаріїв освітлення в межах локальної IoT-інфраструктури.

Колонка *Room 2 Light* реалізує аналогічний до *Room 1 Light* модуль локального контролю освітлення, однак належить до іншого приміщення, де конфігурація джерел світла доповнена окремим індивідуальним

контуром робочого освітлення (рисунок 3.38). Її структура охоплює основні та допоміжні світильники, а також спеціалізоване настільне освітлення, що забезпечує гнучке, енергоефективне й контекстно адаптоване керування світловим середовищем.

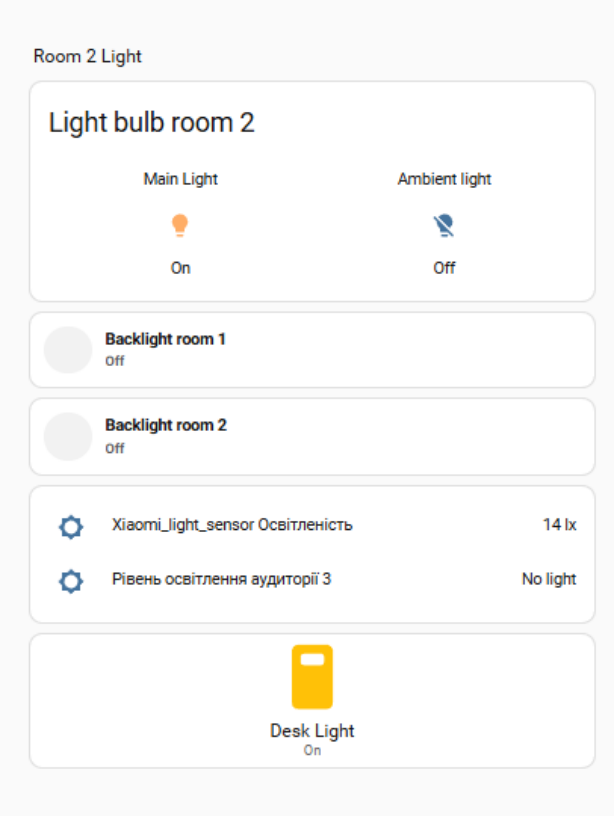

Рис. 3.38: Інтерфейс колонки «Room 2 Light».

У верхній частині інтерфейсу подано блок основного (*Main Light*) та допоміжного (*Ambient Light*) освітлення, які відображаються за допомогою піктограм із кольоровою індикацією поточного стану. У наведеному прикладі основне освітлення активне, тоді як фонова підсвітка вимкнена. Такий підхід дозволяє користувачеві оперативно оцінити поточну конфігурацію освітлення та виконати керування безпосередньо через інтерфейс *Lovelace UI*, без переходу до додаткових меню чи підрозділів.

Наступні елементи - *Backlight room 1* та *Backlight room 2* - відповідають за локальне керування додатковими джерелами підсвітки (наприклад, стельовими або настінними LED-стрічками). Кожен блок може працювати як у ручному режимі, так і в складі сценаріїв автоматизації, синхронізованих із показниками сенсорів освітленості, часовими подіями (світанок, захід сонця) або тригерами присутності користувачів у кімнаті. Аналогічно до колонки *Room 1 Light*, система підтримує можливість регулювання інтенсивності світла та вибору кольору підсвітки, що забезпечує індивідуальне налаштування світлової атмосфери відповідно до поточних умов чи завдань.

Далі розміщено блок сенсорів освітленості, які забезпечують безперервний моніторинг рівня світла в реальному часі. Сенсор *Xiaomi light sensor* надає кількісні показники в люксах (у наведеному прикладі - 14 lx), тоді як параметр *Рівень освітлення аудиторії 3* забезпечує якісну індикацію («No light»). Ці дані використовуються системою для динамічної адаптації освітлення - зокрема, для автоматичного підвищення яскравості при зменшенні природного освітлення або вимкнення надлишкових джерел світла з метою економії енергії.

У нижній частині колонки розташовано окремий елемент *Desk Light*, який відповідає за індивідуальне робоче освітлення - настільну лампу або локальний світильник, що формує оптимальні умови для зорової роботи. Цей модуль є повноцінною складовою системи автоматизації: він може автоматично активуватися при виявленні користувача за робочим місцем, у визначені години або під час переходу системи в режим «робочої активно-

сті». Завдяки інтеграції з *Home Assistant*, *Desk Light* також може реагувати на сценарії, пов'язані з рівнем освітленості, присутністю або навіть розкладом занять.

У цілому колонка *Room 2 Light* виступає розширеним модулем керування освітленням, який поєднує функції моніторингу, динамічного регулювання та персоналізованого освітлення робочої зони. У порівнянні з *Room 1 Light*, вона додатково охоплює контур індивідуального світла (*Desk Light*), що підвищує комфорт користувача та розширює можливості контекстної автоматизації. Такий підхід сприяє підвищенню енергоефективності, покращенню ергономічних характеристик простору та реалізації принципів «розумного освітлення» в межах локальної IoT-системи.

Колонка *Daylight Management* реалізує інтелектуальне керування природним освітленням приміщень шляхом автоматизованого контролю положення жалюзі та аналізу показників освітленості, отриманих від сенсорів, інтегрованих у систему *Home Assistant* (рисунок 3.39). Вона поєднує аналітичні та виконавчі модулі, що дозволяє адаптивно регулювати проникнення сонячного світла залежно від часу доби, поточного рівня освітленості, а також астрономічних подій, таких як світанок і захід сонця.

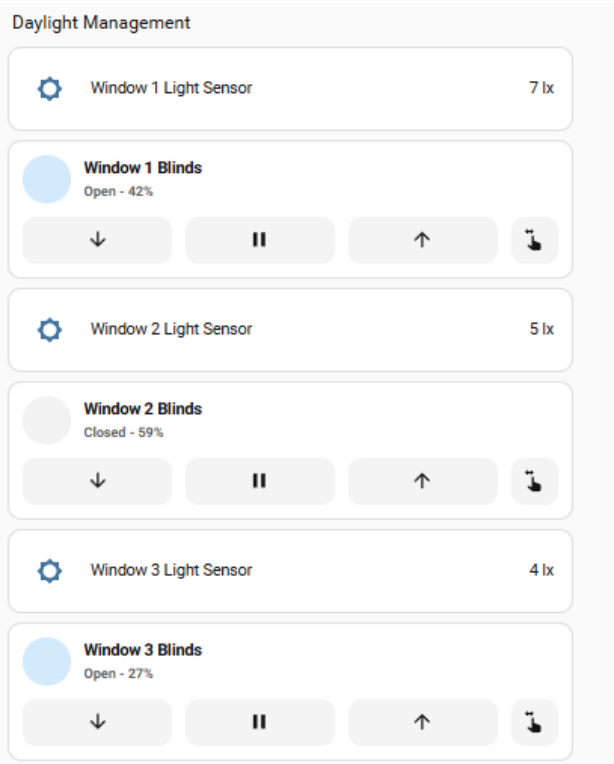

Рис. 3.39: Інтерфейс колонки «Daylight Management».

Структурно цей блок передбачає три незалежні канали керування для різних віконних зон (*Window 1*, *Window 2*, *Window 3*). Кожен канал включає сенсор освітленості та відповідний виконавчий механізм жалюзі. Сенсори (*Window 1/2/3 Light Sensor*) вимірюють рівень освітлення в люксах (7 lx, 5 lx, 4 lx відповідно) та передають дані до ядра системи для обробки в режимі реального часу. На основі цих показників виконується динамічна зміна положення жалюзі з метою досягнення комфортного рівня освітленості всередині приміщення.

Кожен блок керування жалюзі (*Window 1–3 Blinds*) містить інтерфейсні елементи для ручного, напівавтоматичного та повністю автоматизованого режимів роботи. У ручному режимі користувач може змінювати положення полотна за допомогою кнопок підняття, опускання або зупинки, тоді як автоматичний режим базується на сценаріях *Home Assistant* з урахуванням часу сходу/заходу сонця, поточного стану освітленості та встановлених користувацьких пріоритетів. Логіка автоматизації побудована на умовах типу «якщо–то», зокрема:

- якщо рівень освітлення перевищує 15 lx - система поступово знижує жалюзі до 50 %;
- якщо зафіксовано захід сонця - виконується повне закриття жалюзі з метою енергозбереження;

- якщо освітленість зменшується нижче 5 lx - система частково піднімає жалюзі, підтримуючи мінімально необхідний рівень природного світла.

Такий підхід забезпечує стабільний світловий режим у приміщеннях, знижує навантаження на систему штучного освітлення та сприяє загальній енергоефективності. У поєднанні з даними про сонячну активність (отриманими з модуля *Sun integration*) модуль *Daylight Management* створює умови для автоматизованого керування освітленням відповідно до природного добового циклу. Крім того, система може враховувати розташування робочих місць і поточні погодні умови, що дозволяє запобігати надмірному освітленню або появі відблисків у робочих зонах.

У результаті реалізується концепція «адаптивного освітлення», спрямована на підвищення комфорту, продуктивності та візуальної ергономіки у навчальному чи робочому середовищі. Інтеграція *Daylight Management* із рештою модулів панелі «Light» забезпечує узгоджену взаємодію між природним та штучним освітленням, формуючи єдину інтелектуальну систему керування світловим середовищем об'єкта.

3.2.4. Інформаційна панель «System».

Інформаційна панель системного моніторингу (*System Dashboard*) виконує функцію централізованого інтерфейсу технічного контролю локального серверного середовища, мережевої інфраструктури та програмних компонентів системи автоматизації на базі *Home Assistant* (рисунок 3.40). Її основним призначенням є забезпечення комплексної діагностики стану апаратних ресурсів, моніторингу мережевих підключень, оцінки стабільності електроживлення, а також контролю актуальності оновлень і процесів резервного копіювання системи. Програмний код налаштування інтерфейсу можна знайти у підрозділі Д.1

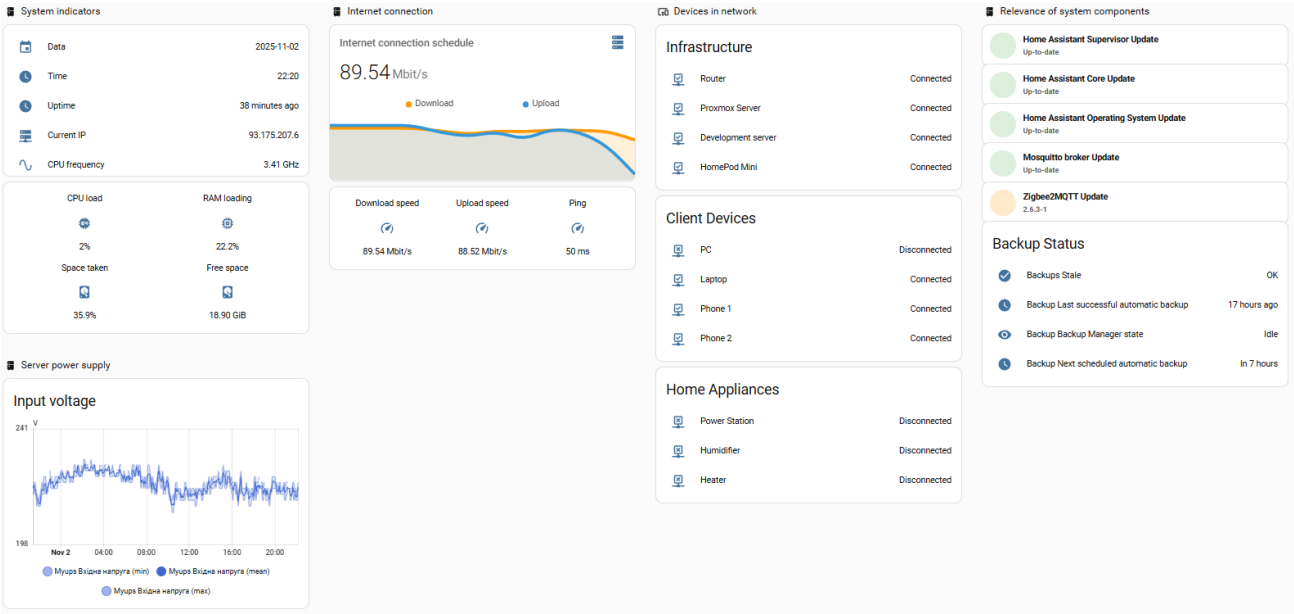

Рис. 3.40: Загальний вигляд системної панелі моніторингу.

Архітектура панелі побудована за модульним принципом і складається з кількох логічно впорядкованих колонок, кожна з яких реалізує окрему діагностичну або сервісну функцію. На рівні єдиного інтерфейсу об'єднано телеметричні дані апаратного рівня (стан процесора, оперативної пам'яті, мережевих інтерфейсів, параметри напруги живлення) та програмного рівня (актуальність компонентів, статус резервного копіювання, доступність пристроїв у локальній мережі).

Перший сегмент панелі відповідає за моніторинг базових системних параметрів, зокрема дати, часу, тривалості безперервної роботи (Uptime), завантаження центрального процесора та використання оперативної пам'яті. Наступні елементи відображають стабільність інтернет-з'єднання та пропускну здатність мережевого каналу, що дає змогу контролювати якість зв'язку між локальним сервером і зовнішніми ресурсами. Окремий блок

присвячено візуалізації стану підключених пристроїв у локальній мережі, згрупованих за категоріями: інфраструктурні вузли, клієнтські пристрої та побутове обладнання. Такий підхід забезпечує наочне уявлення про активність мережевих елементів і дає змогу оперативно діагностувати втрату зв'язку або відмову окремих компонентів.

Додатково реалізовано моніторинг електроживлення серверної частини, що відображає коливання вхідної напруги та дає змогу аналізувати її стабільність у часі. Для систем, які працюють у режимі цілодобового навантаження, цей параметр є критично важливим з точки зору надійності та безперервності функціонування. У правій частині панелі розміщено блок контролю актуальності оновлень ключових компонентів середовища *Home Assistant* та статусу резервного копіювання. Користувач може відстежувати час останнього створення резервної копії, запланований час наступної, а також поточний стан основних програмних модулів системи.

Завдяки інтеграції телеметрії, графічних віджетів і механізмів зворотного зв'язку інформаційна панель забезпечує оперативне виявлення відхилень у роботі серверного середовища, підтримує ухвалення технічних рішень у режимі реального часу та сприяє підвищенню загальної стабільності й безпеки системи. Її використання формує основу для аналітичного моніторингу стану локальної IoT-інфраструктури та забезпечує прозоре керування всіма технічними підсистемами.Логіку автоматизації описано в додатку Д.2.

Колонка *System Indicators and Power Supply* виконує роль центрального діагностичного блоку, який забезпечує моніторинг ключових параметрів серверного середовища, стану обчислювальних ресурсів і стабільності електроживлення (рисунок 3.41). Вона поєднує інформацію апаратного та мережевого рівнів, надаючи оператору узгоджену картину поточного стану системи в режимі реального часу.

У верхній частині колонки розміщено інформаційний блок *System Indicators*, який відображає поточну дату, час, тривалість безперервної роботи (Uptime), публічну IP-адресу сервера та частоту центрального процесора. Сукупність цих параметрів дає змогу оцінити стабільність роботи системи, наявність зовнішнього мережевого підключення та відповідність фактичних характеристик заданому режиму експлуатації. Нижче розташовано секцію моніторингу ресурсів, яка містить показники завантаження процесора (*CPU load*), використання оперативної пам'яті (*RAM loading*), а також зайнятого й вільного дискового простору. Такий формат візуалізації забезпечує можливість відстежувати продуктивність системи та вчасно виявляти потенційні перевантаження, що є критично важливим для стабільності серверного середовища *Home Assistant* і пов'язаних сервісів.

Другий логічний блок - *Server Power Supply* - реалізує моніторинг параметрів живлення сервера. Графічний віджет *Input Voltage* відображає коливання вхідної напруги у часовій динаміці (мінімальне, середнє та максимальне значення), що дає змогу проводити енергетичну діагностику й аналізувати стабільність електроживлення. Це безпосередньо впливає на надійність безперервної роботи системи, особливо за умов використання джерел безперебійного живлення (UPS) або автономних енергомодулів у локальному середовищі.

Отже, колонка *System Indicators and Power Supply* є базовим елементом технічного моніторингу: вона об'єднує апаратні, мережеві й енергетичні показники, забезпечуючи комплексну оцінку поточного стану інфраструктури. Її функціональність спрямована на підтримання стабільності, оптимізацію навантаження та раннє виявлення відхилень у роботі серверної системи.

Колонка *Internet Connection* виконує функцію моніторингу мережевих характеристик і стабільності інтернет-з'єднання серверної системи. Її структура включає два основні інформаційні блоки: динамічний графік швидкостей передавання даних та аналітичний модуль поточних показників продуктивності мережі (рисунок 3.42).

У верхній частині колонки відображається графік зміни швидкості завантаження (*Download*) та вивантаження (*Upload*) даних у режимі реального часу. Візуалізація побудована за допомогою реактивних компонентів *Lovelace UI*, що забезпечують автоматичне оновлення даних без перезавантаження сторінки. Такий підхід дозволяє здійснювати безперервний контроль якості мережевого каналу, виявляти короткочасні стрибки швидкості або епізоди зниження стабільності з'єднання.

У нижньому блоці наведено актуальні числові значення ключових параметрів мережі: швидкість завантаження (89,54 Мбіт/с), швидкість вивантаження (88,52 Мбіт/с) та затримку сигналу (Ping - 50 мс). Ці дані формуються шляхом періодичних перевірок з'єднання із зовнішніми сервісами або шлюзом маршрутизатора, що дає змогу оцінювати ефективність використання пропускної здатності каналу. Колонка *Internet Connection*

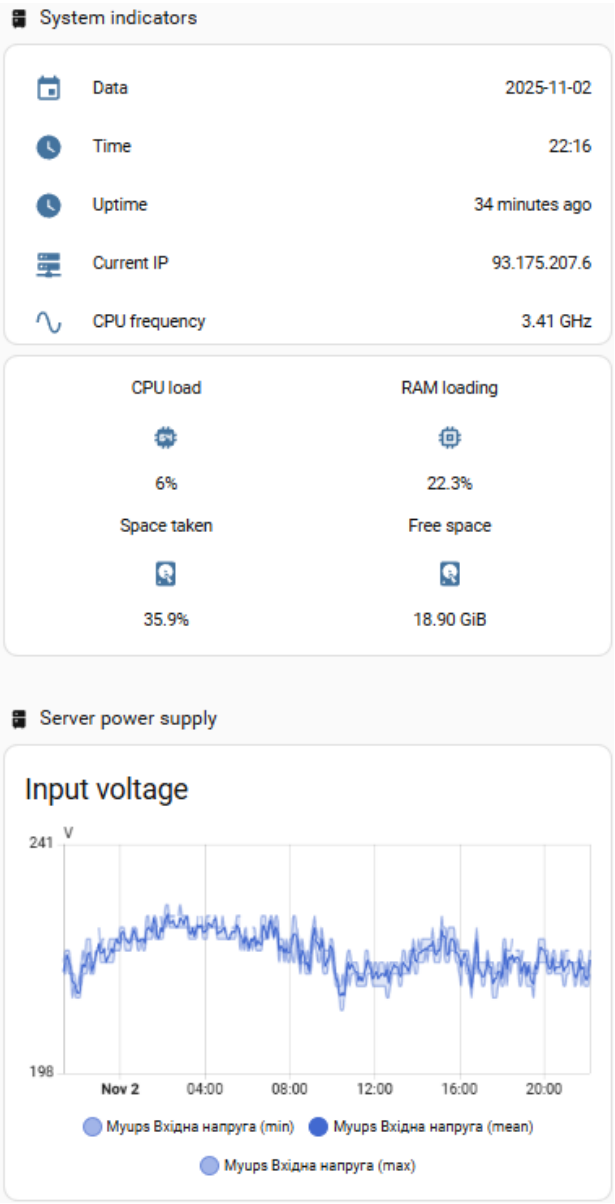

Рис. 3.41: Інтерфейс колонки «System Indicators and Power Supply».

має не лише інформативну, а й діагностичну функцію: вона дозволяє оперативно оцінювати стан мережевої інфраструктури, виявляти потенційні проблеми в роботі інтернет-провайдера або локального маршрутизатора та забезпечувати своєчасне реагування системи автоматизації на зміну умов зв'язку (наприклад, перемикання на резервний канал).

Таким чином, цей модуль забезпечує неперервний моніторинг параметрів мережевого середовища, підвищує стабільність роботи системи та сприяє збереженню її автономності навіть за умов коливань зовнішнього інтернет-з'єднання.

Колонка *Devices in Network* реалізує функцію моніторингу активності та доступності пристроїв у локальній мережі. Вона побудована за принципом структурної класифікації обладнання за типами - *Infrastructure*, *Client Devices* та *Home Appliances*, що забезпечує швидку орієнтацію користувача в топології мережі та дозволяє здійснювати оперативну діагностику її стану (рисунок 3.43).

Моніторинг стану підключення виконується за допомогою періодичних ICMP-запитів (*ping*), що дають змогу визначити факт доступності або недоступності кожного вузла локальної мережі. Отримані результати автоматично трансформуються у статуси *Connected* або *Disconnected*, забезпечуючи користувача актуальною інформацією про стабільність роботи мережевої інфраструктури.

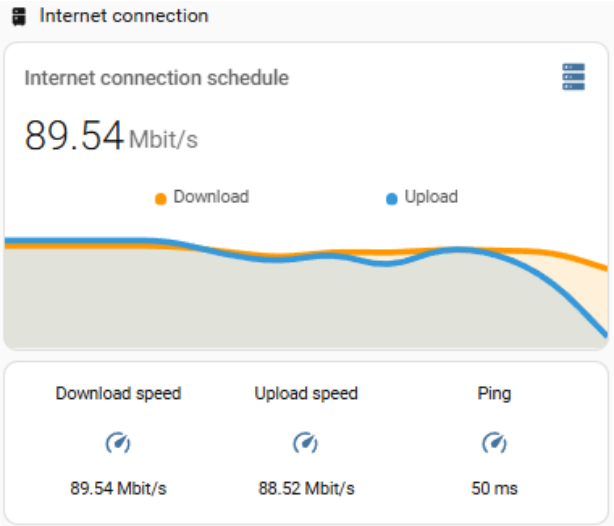

Рис. 3.42: Інтерфейс колонки «Internet Connection».

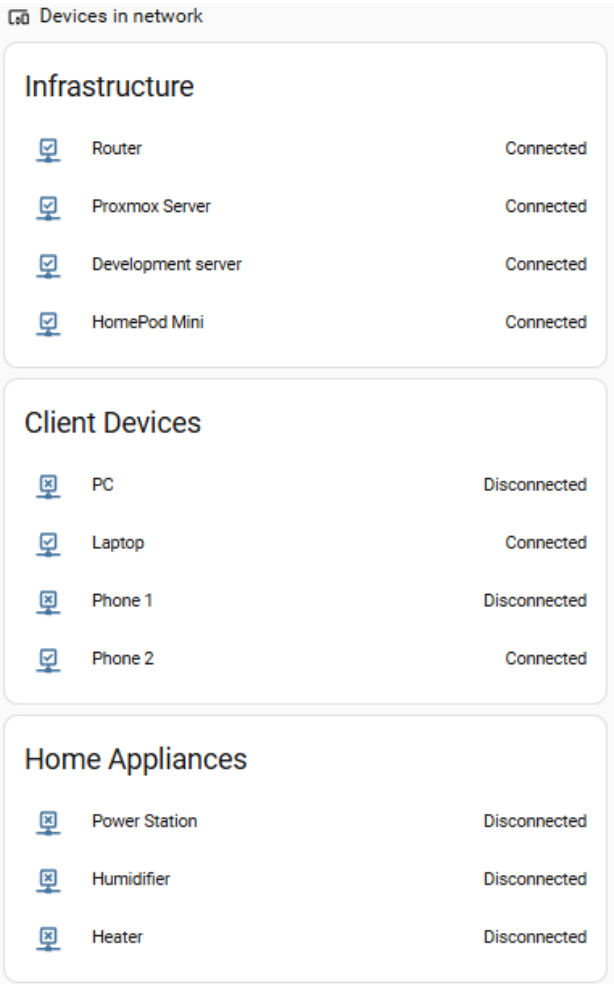

Рис. 3.43: Інтерфейс колонки «Devices in Network».

Блок *Infrastructure* містить ключові елементи серверного та мережевого рівня - маршрутизатор, віртуалізаційну платформу Proxmox, серверні вузли та мультимедійний пристрій HomePod Mini. Стабільна доступність цих компонентів свідчить про коректне функціонування базової інфраструктури системи автоматизації та відсутність критичних збоїв.

Розділ *Client Devices* охоплює персональні пристрої користувачів - настільний ПК, ноутбук та мобільні телефони. Статуси активності цих пристроїв відображають поточну взаємодію користувачів із системою і можуть бути використані як тригери у сценаріях автоматизації, наприклад, для визначення присутності мешканців або активації відповідних профілів роботи.

Окремий блок *Home Appliances* призначено для моніторингу побутових пристроїв - енергостанції, зволожувача повітря, обігрівача тощо. Статуси підключення дозволяють не лише оцінювати наявність живлення та доступність пристрою, а й створювати автоматизовані повідомлення у разі відключення чи виявлення аномалій у роботі.

Таким чином, колонка *Devices in Network* забезпечує централізований контроль за станом усіх вузлів локальної IoT-інфраструктури. Її функціональність поєднує моніторинг, діагностику та аналіз мережевої активності, що підвищує загальну стійкість системи й скорочує час реагування на відмови або втрату зв'язку.

Колонка *Relevance of System Components* виконує функцію контролю актуальності та цілісності ключових модулів програмного середовища *Home Assistant*, а також моніторингу стану резервного копіювання системи (рисунок 3.44). Її основне призначення полягає у підтриманні стабільності, відмовостійкості та здатності системи до оперативного відновлення у разі збоїв.

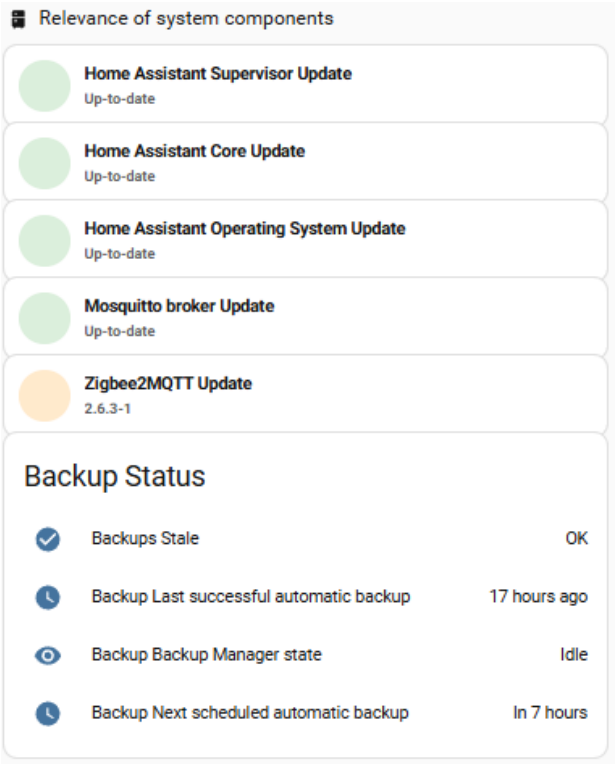

Рис. 3.44: Інтерфейс колонки «Relevance of System Components».

Верхній блок відображає поточний статус оновлень основних компонентів системи - *Home Assistant Supervisor*, *Core*, *Operating System*, брокера повідомлень *Mosquitto* та шлюзу *Zigbee2MQTT*. Для кожного елемента позначено його встановлену версію, а також ступінь відповідності найновішому офіційному релізу. Візуальні індикатори (зелені або жовті маркери) дозволяють швидко оцінити стан оновлення без додаткових переходів у меню налаштувань. Така модель контролю зменшує ризики використання застарілих компонентів, підвищує рівень кібербезпеки та забезпечує сумісність між підсистемами.

Нижній блок - *Backup Status* - відповідає за моніторинг процесів резервного копіювання. Він відображає дані щодо часу останнього успішного бекапу (у наведеному прикладі - 17 годин тому), поточного стану менеджера резервування та інтервалу до наступного запланованого збереження системи. Додатковий індикатор *Backups Stale* сигналізує про актуальність і цілісність існуючих резервних копій, що дозволяє своєчасно реагувати на можливі збої в системі збереження.

Ключовою особливістю цього модуля є його автономність: оновлення та створення резервних копій виконуються локально, без потреби звернення до зовнішніх хмарних сервісів. Це зменшує залежність від зовнішньої інфраструктури, підвищує стійкість системи та забезпечує прозорий контроль над усіма критично важливими компонентами середовища *Home Assistant*.

Таким чином, колонка *Relevance of System Components* виконує роль діагностичного центру, що поєднує аналіз актуальності компонентів, контроль резервного копіювання та оцінку стабільності системи. Її інтеграція сприяє підтриманню безперервності роботи, інформаційної безпеки та довготривалої надійності експлуатації локальної інфраструктури автоматизації.

3.2.5. Інформаційна система «Battery».

У системі локальної автоматизації середовища на базі Home Assistant панель *Battery* виконує функцію комплексного моніторингу енергетичних процесів, забезпечуючи відображення показників електроспоживання, стану акумуляторних систем, параметрів напруги та рівня навантаження в реальному часі. Вона інтегрує дані з широкого спектра джерел - сенсорів напруги, розумних розеток, реле споживання, інверторів і резервних енергостанцій - формуючи цілісну картину енергетичного балансу в межах локальної IoT-інфраструктури. Програмний код налаштування інтерфейсу можна знайти у підрозділі Е.1

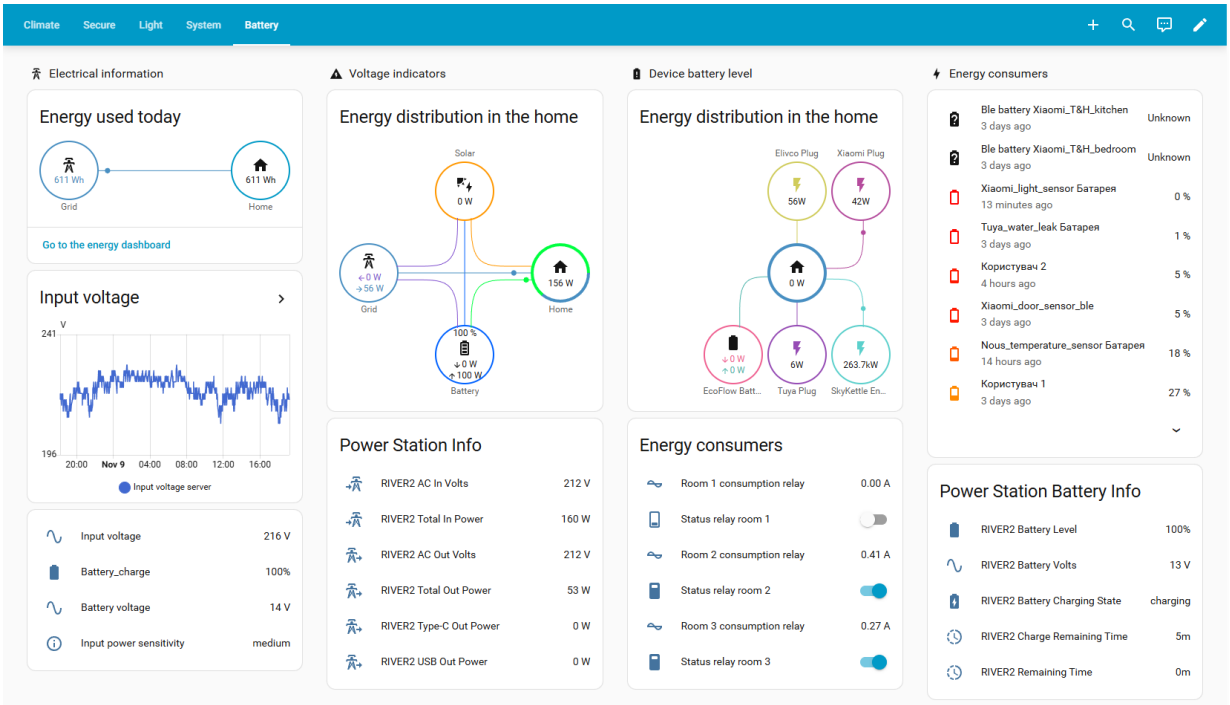

Рис. 3.45: Інтерфейс інформаційної панелі «Battery».

Структура панелі реалізована у вигляді кількох взаємопов'язаних колонок, що охоплюють основні напрямки енергомоніторингу: загальне споживання електроенергії, розподіл енергії в домі, показники напруги, рівень заряду пристроїв та контроль енергоспоживачів. Усі елементи синхронізуються через локальну шину даних і відображають актуальні вимірювання у форматі динамічних графіків, таблиць і схем потоків енергії. Логіку автоматизації описано в додатку Е.2

Панель *Battery* дає змогу виявляти аномалії у споживанні, контролювати роботу автономних джерел живлення (зокрема портативних енергостанцій типу EcoFlow) та оцінювати ефективність розподілу навантажень між зонами або окремими пристроями. Вона також підтримує сценарії автоматизації, що реагують на зміну напруги, рівень заряду чи перевищення споживання - наприклад, відключення неключових пристроїв при переході на живлення від батареї. Завдяки інтегрованому підходу, панель *Battery* формує єдиний аналітичний центр контролю енергетичного стану об'єкта. Вона сприяє підвищенню енергоефективності, стабільності живлення та

надійності всієї локальної IoT-екосистеми, забезпечуючи прозорість енергопроцесів і можливість їх гнучкого управління (рисунок 3.45).

Колонка *Electrical information* виконує роль базового індикатора енергетичного стану системи автоматизації. Вона забезпечує візуалізацію основних параметрів споживання та постачання електроенергії, а також даних щодо стабільності живлення серверної інфраструктури. Завдяки цьому користувач має змогу не лише оцінювати поточний стан енергоспоживання, але й здійснювати аналіз тенденцій, що безпосередньо впливають на енергоефективність і надійність роботи всієї локальної IoT-мережі (рисунок 3.46).

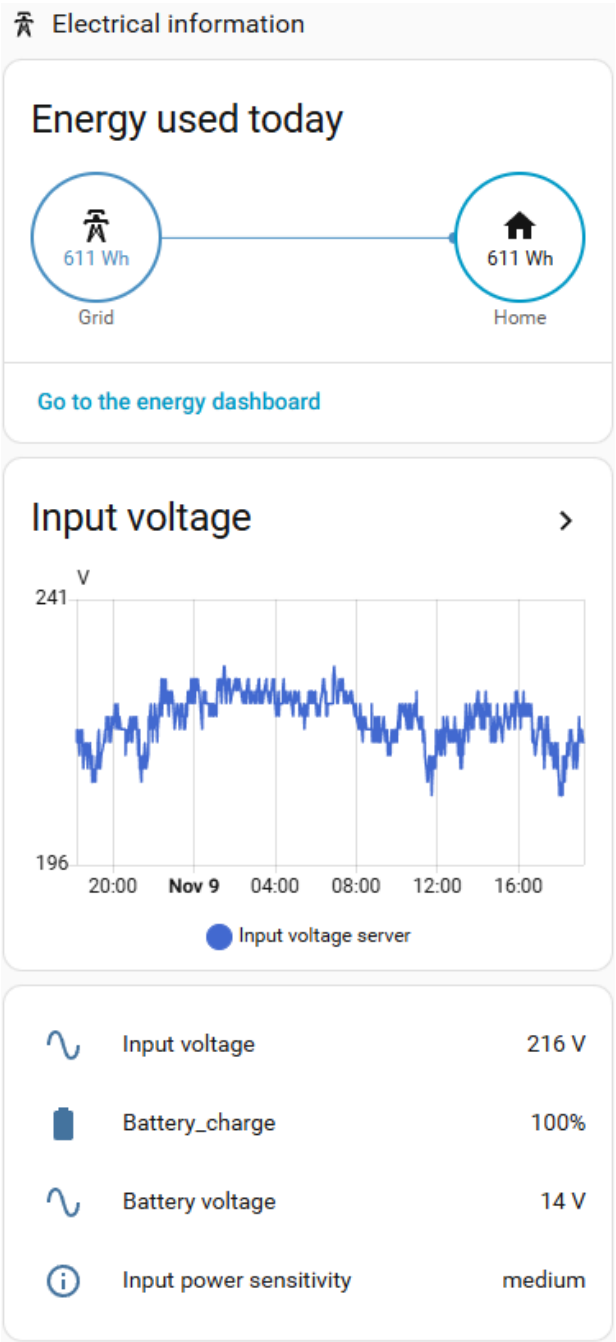

Рис. 3.46: Інтерфейс колонки «Electrical information».

У верхній частині колонки відображено узагальнену добову статистику енергоспоживання у форматі зв'язку між двома вузлами - мережею живлення (*Grid*) та споживачем (*Home*). Цей блок дає змогу оцінити сумарну кількість енергії, спожиту за поточну добу, що є важливим параметром при аналізі стабільності енергопостачання та побудові профілю навантаження у динаміці. Візуалізація реалізована у формі лінійного зв'язку між

точками постачання та споживання, що полегшує інтерпретацію даних навіть за умови значних коливань рівнів споживання протягом доби.

Нижче розміщено графічний блок моніторингу *Input voltage*, який відображає коливання вхідної напруги у реальному часі. Цей графік дає можливість виявляти короткочасні просідання або стрибки напруги, що можуть свідчити про нестабільність електромережі чи перевантаження лінії. Характер кривої (плавний чи фрагментований) дає змогу оперативно оцінювати якість електропостачання, що є критично важливим для систем з безперервним циклом роботи, таких як сервер Home Assistant.

У нижній частині колонки розташовані детальні технічні індикатори системи безперебійного живлення (UPS), які дозволяють відстежувати не лише стан акумулятора, але й параметри мережевої чутливості. До них належать:

- *Input voltage* - поточне значення вхідної напруги, що надходить на UPS;
- *Battery charge* - рівень заряду акумуляторної батареї у відсотках;
- *Battery voltage* - фактична напруга живлення батареї, що визначає її стан і працездатність;
- *Input power sensitivity* - встановлений режим чутливості системи до коливань мережевої напруги (у наведеному прикладі - середній рівень).

Комплексна інтерпретація цих показників дає змогу здійснювати моніторинг якості енергопостачання, контролювати заряд та напругу резервного джерела живлення, а також вчасно виявляти відхилення у роботі мережі або UPS. Завдяки інтеграції з аналітичними механізмами Home Assistant, ці дані можуть використовуватися у сценаріях автоматизації, наприклад, для сповіщення про низький заряд батареї, перемикання у режим енергозбереження або планування часу роботи обладнання під час автономного живлення.

Таким чином, колонка *Electrical information* виконує функцію енергетичного моніторингового модуля, який поєднує вимірювальні, діагностичні та аналітичні можливості. Її застосування підвищує стабільність функціонування серверного вузла, забезпечує контроль за якістю електропостачання та сприяє реалізації концепції енергоефективного «розумного середовища» у межах локальної IoT-інфраструктури.

Колонка *Voltage indicators* відображає ключові параметри розподілу та якості електроживлення у межах локальної енергетичної інфраструктури. Її основна функція полягає у візуалізації потоків споживання та передавання енергії між основними джерелами живлення - мережею, сонячними панелями, акумуляторною батареєю та домашніми споживачами (рисунок 3.47). Такий підхід дозволяє користувачеві оперативно оцінювати баланс між джерелами вироблення енергії та навантаженням у системі, а також визначати ефективність роботи резервних модулів.

У верхній частині колонки наведена діаграма енергетичних потоків *Energy distribution in the home*, яка демонструє взаємодію між чотирма основними вузлами: *Grid* (мережа), *Solar* (сонячна енергія), *Battery* (акумуляторна станція) та *Home* (споживач). Кожен напрямок на діаграмі відображає поточну потужність у ватах, що дає змогу відстежувати в реальному часі як споживання, так і заряд або розряд акумуляторів. Наприклад, при підключенні до сонячних панелей користувач може бачити, яку частину енергії подано в мережу, а яку - використано для власних потреб або заряджання батареї.

Другий блок *Power Station Info* містить детальні технічні параметри зарядної станції (у даному випадку - моделі *RIVER2*). Серед показників відображаються:

- *AC In Volts* - вхідна напруга змінного струму;
- *Total In Power* - сумарна споживана потужність;
- *AC Out Volts* - вихідна напруга;
- *Total Out Power* - сумарна потужність на виході;
- *Type-C Out Power* та *USB Out Power* - потужність, що подається на відповідні порти.

Завдяки такій структурі користувач отримує інтегровану картину розподілу енергії між різними джерелами, у тому числі мережею, сонячними панелями та акумуляторними станціями. Ці дані дають змогу оцінювати ефективність енергоспоживання, визначати можливість автономної роботи у разі відключення основного живлення, а також аналізувати рівень навантаження на окремі вузли системи.

Колонка *Voltage indicators* є важливим інструментом у межах панелі «Battery», оскільки забезпечує ком-

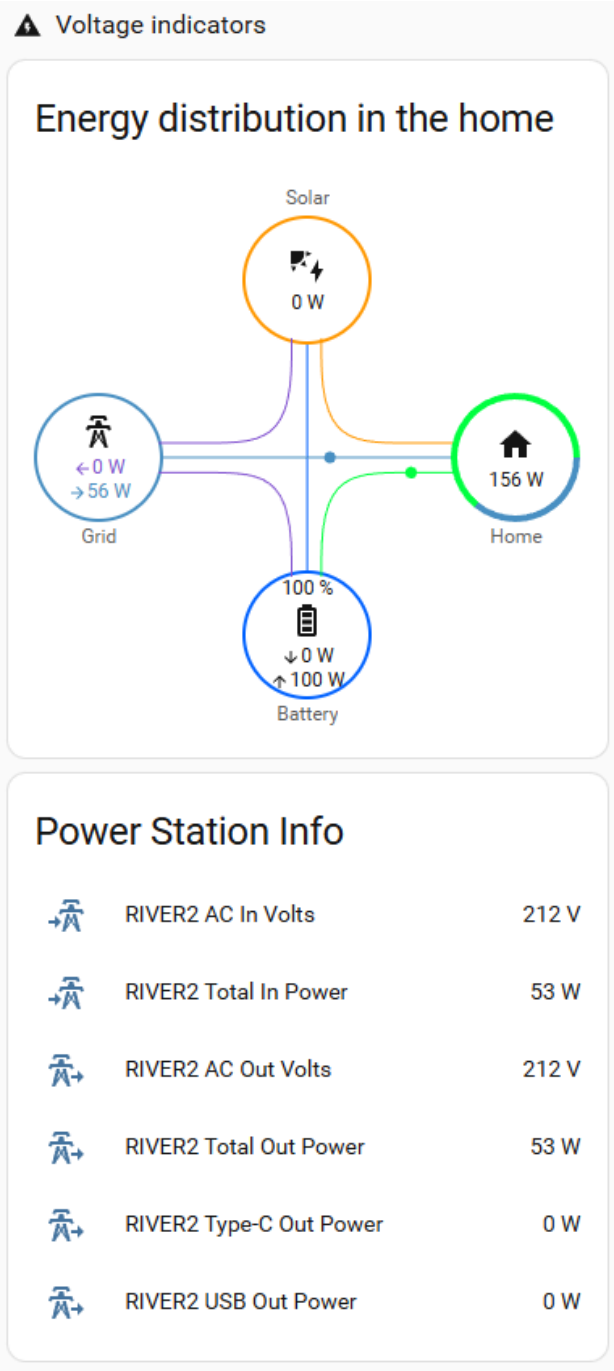

Рис. 3.47: Інтерфейс колонки «Voltage indicators».

плексний моніторинг енергетичних потоків, поєднуючи технічні дані та графічну інтерпретацію у зручному форматі. Вона сприяє оптимізації розподілу навантаження, підвищенню стабільності електропостачання та реалізації принципів сталого енергокерування в межах інтелектуальної IoT-інфраструктури.

Колонка *Energy consumers* відображає поточний стан споживання електроенергії окремими пристроями у системі та надає зведену візуалізацію енергетичних потоків усередині локальної інфраструктури (рисунок 3.48). Її головна мета полягає у забезпеченні швидкого аналізу навантаження та визначенні активних споживачів у реальному часі, що дозволяє користувачеві ефективно управляти енергоресурсами і контролювати роботу підключених пристроїв.

У верхній частині колонки представлено блок *Energy distribution in the home*, що візуалізує взаємозв'язки між активними споживачами енергії, зокрема розумними розетками, електроприладами та додатковими елементами живлення. Кожен об'єкт відображено у вигляді окремого вузла з індикацією поточного споживання

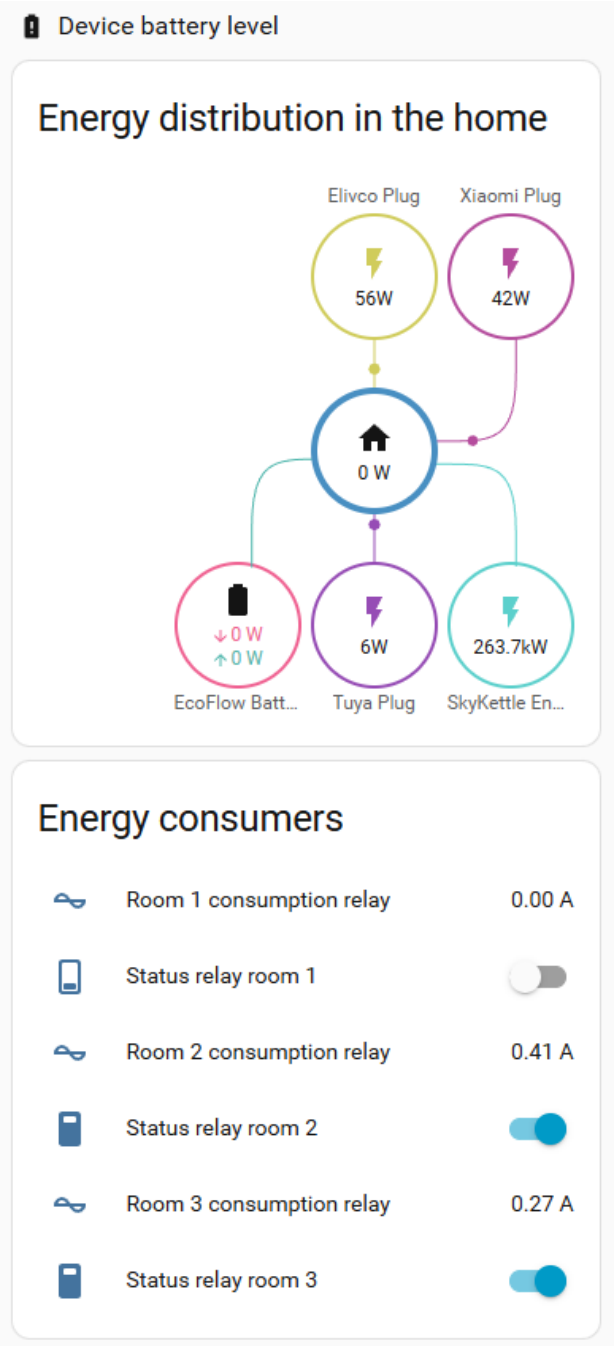

Рис. 3.48: Інтерфейс колонки «Energy consumers».

у ватах, що дає змогу виявити найбільш енергоємні компоненти системи. Такий підхід забезпечує не лише оперативний моніторинг, а й формування бази даних для подальшого енергетичного аналізу, прогнозування навантаження або оптимізації режимів роботи.

Нижче розташовано секцію *Energy consumers*, де наведено перелік споживачів за зонами або кімнатами (*Room 1–3*). Для кожного з них подається значення струму навантаження в амперах, що відображає інтенсивність споживання на даний момент. Поруч розташовані елементи керування реле (*Status relay*), які дозволяють вручну вмикати або вимикати живлення для відповідної лінії. Така інтеграція забезпечує зручне поєднання моніторингу та управління, оскільки користувач бачить не лише поточну потужність споживання, але й може оперативно втрутитися у роботу системи.

Таким чином, колонка *Energy consumers* виконує функції аналітичного центру моніторингу енергоспоживання на рівні окремих пристроїв. Вона поєднує візуалізацію енергетичних потоків, аналітичні показники на-

вантаження та інструменти прямого керування, що дозволяє досягти високого рівня прозорості, ефективності та адаптивності в управлінні енергетичною інфраструктурою розумного середовища.

Колонка *Device battery level* виконує функцію централізованого моніторингу стану автономних пристроїв і зарядної станції, забезпечуючи контроль енергетичного балансу в системі (рисунок 3.49). Вона об'єднує інформацію про рівень заряду елементів живлення бездротових сенсорів, а також надає актуальні параметри роботи стаціонарної енергетичної станції, яка виступає джерелом резервного живлення для серверної частини IoT-інфраструктури.

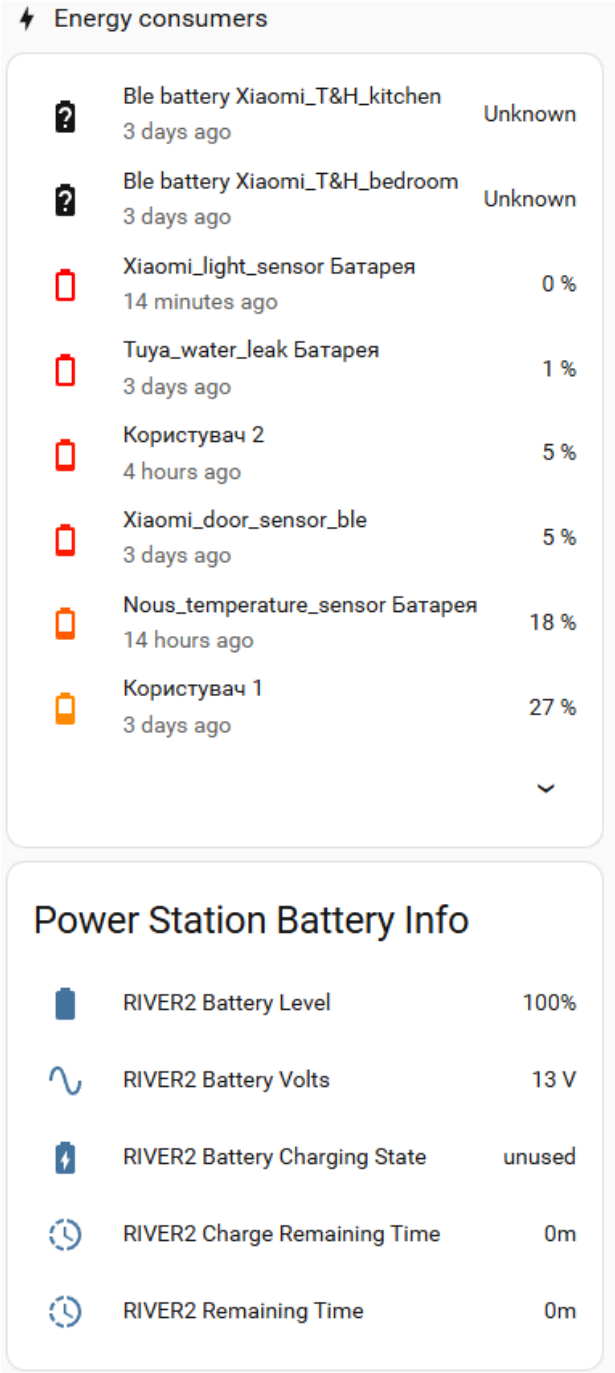

Рис. 3.49: Інтерфейс колонки «Device battery level».

У верхній частині колонки відображено перелік автономних сенсорів та BLE-пристроїв, для яких система в реальному часі контролює рівень заряду вбудованих акумуляторів. Серед них можуть бути датчики температури, вологості, витоку води, освітленості, дверні сенсори, а також інші периферійні пристрої, інтегровані через

Zigbee- або Bluetooth-шину. Для кожного елемента зазначено останній час оновлення даних і відсотковий рівень заряду, що дозволяє своєчасно виявляти пристрої, які потребують підзаряджання або заміни елементів живлення. Такий підхід підвищує надійність системи, запобігаючи відмовам критичних сенсорів, що забезпечують моніторинг середовища, безпеку чи контроль мікроклімату.

Система автоматично виділяє пристрої зі зниженим зарядом, використовуючи кольорову індикацію та попереджувальні піктограми. Це дає змогу швидко зорієнтуватися у стані енергозабезпечення без необхідності додаткової ручної діагностики. Інтеграція з механізмами Home Assistant дозволяє автоматизувати повідомлення про низький заряд - зокрема, надсилати сповіщення до мобільного застосунку або на електронну пошту адміністратора.

Нижній блок *Power Station Battery Info* відображає ключові параметри зарядної станції *RIVER2*, яка використовується як джерело резервного живлення для серверного вузла системи. Тут подано інформацію про рівень заряду акумулятора у відсотках, поточний вольтаж, стан зарядки, а також розрахункові значення часу заряджання та залишкової автономної роботи під навантаженням. Ці параметри дають змогу оцінити готовність системи до переходу в автономний режим у разі відключення основного живлення.

Дані зчитуються через API зарядної станції та локально обробляються Home Assistant без використання хмарних сервісів, що забезпечує високу швидкодію та стабільність передачі інформації. На основі цих показників можуть формуватися автоматизації - наприклад, відключення другорядних пристроїв при зниженні заряду до критичного рівня або сповіщення користувача про завершення циклу зарядки.

Таким чином, колонка *Device battery level* виконує комплексну аналітичну функцію, поєднуючи контроль стану автономних сенсорів та моніторинг роботи резервної енергетичної системи. Вона є ключовим елементом забезпечення енергетичної стабільності локальної IoT-мережі, сприяє оптимізації експлуатаційних процесів і гарантує безперервність роботи автоматизованого середовища.

3.3. Висновки по розділу 3

1. У розділі виконано практичну реалізацію локальної IoT-системи на основі платформи Home Assistant, що включає конфігурацію головної панелі керування, навігаційних елементів та базових модулів взаємодії з підсистемами. Така реалізація забезпечує не лише відтворення теоретичної архітектури, описаної у попередньому розділі, але й надає можливість оцінити її працездатність у реальних умовах експлуатації. Це дозволяє підтвердити життєздатність і практичну придатність запропонованої архітектурної моделі.
2. Деталізовано побудову інтерфейсу користувача, зокрема структури бічного меню, панелей керування та механізмів персоналізації. Такий підхід дозволяє створити інтуїтивно зрозуміле середовище, яке мінімізує навчальне навантаження для користувача та забезпечує ефективну взаємодію з великою кількістю сенсорів і пристроїв. Завдяки цьому підвищується керованість, прозорість виконуваних дій та швидкість доступу до критично важливих функцій системи.
3. Представлено функціональні сторінки підсистем (клімат, освітлення, енергомоніторинг, безпека, керування денним освітленням) та проведено аналіз їх роботи в режимі реального часу. Це дозволяє оцінити здатність системи адаптувати поведінку до зміни середовищних параметрів та контекстних умов. Застосування сенсорних даних у циклі реального часу забезпечує формування автономних сценаріїв, що зменшує потребу у втручанні користувача і підвищує енергоефективність та комфорт.
4. Описано інтеграцію виконавчих модулів — контурів освітлення, жалюзі, HVAC-компонентів, датчиків присутності та механізмів керування. Реалізовані інструменти підтверджують можливість використання гетерогенної інфраструктури (Wi-Fi, Zigbee, MQTT) у межах єдиної системи. Це забезпечує масштабованість, сумісність і можливість подальшого розширення без порушення цілісності архітектури. Таким чином, система може бути адаптована до зростання кількості пристроїв або розширення функціональних потреб.
5. Узагальнено, що побудована практична реалізація демонструє ефективність локально орієнтованої

IoT-системи з погляду автономності, безпеки, керованості та користувацької інтерпретованості. Це дозволяє застосовувати систему у професійних та інституційних сценаріях, де критично важливими є контрольованість інфраструктури, конфіденційність даних та гнучкість у конфігурації сценаріїв. Отримані результати підтверджують, що запропонована модель може бути використана як основа для подальшого розвитку автоматизованих рішень для розумних офісних та освітніх просторів.

РОЗДІЛ 4
ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ФУНКЦІОНАЛЬНОСТІ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЛОКАЛЬНОЇ IOT-СИСТЕМИ НА ОСНОВІ HOME ASSISTANT

У цьому розділі наведено результати експериментальної перевірки функціональності розробленої локальної IoT-системи та оцінки її ефективності. Для цього застосовано методику оцінювання, розроблену у другому розділі, яка ґрунтується на багатокритеріальному підході та охоплює технічні, експлуатаційні й безпекові характеристики системи. Отримані результати зіставлено з існуючими аналогічними рішеннями, що дало змогу визначити переваги запропонованої архітектури, а також окреслити потенційні обмеження її застосування. Таким чином забезпечено комплексну валідацію функціональної моделі та підтверджено практичну значущість представленого підходу.

Особливу увагу приділено вибору, конфігурації та порівняльному аналізу комунікаційних пристроїв, що забезпечують взаємодію елементів мережі IoT, зокрема координаторів та сенсорних вузлів стандарту Zigbee. У процесі експериментальної експлуатації виконано оцінювання їхньої стабільності, енергоспоживання, дальності зв'язку та сумісності з екосистемою Home Assistant. Окремо розглянуто процедури налаштування мережі, реєстрації пристроїв, управління топологією та моніторингу ключових комунікаційних параметрів у режимі реального часу.

У розділі також систематизовано виявлені технічні виклики, що виникли під час інсталяції й експлуатації системи, зокрема прояви бездротової інтерференції, обмеження радіусів покриття, специфіку роботи за наявності великої кількості підключень та особливості розміщення координаторів. Для кожного з виявлених викликів наведено практичні підходи до їх подолання та оптимізації роботи мережевої інфраструктури.

Практичні результати, отримані під час експлуатації системи, підтверджують працездатність і ефективність запропонованої архітектури, а також дозволяють сформулювати рекомендації щодо її подальшої масштабованості, оптимізації використання ресурсів і підвищення надійності локальних IoT-мереж у навчальних та офісних умовах.

Основні результати, представлені у цьому розділі, опубліковані у роботах автора [127, 128]

4.1. Деталі розгортання та налаштування в реальних умовах

Для перевірки системи в реальних умовах було розгорнуто та протестовано локальний сервер автоматизації на базі Home Assistant у трьох окремих офісних кімнатах та одній кімнаті для обслуговування, загальною площею приблизно 50 квадратних метрів (3 офіси $\times$ 15 м^2 та зона обслуговування 1 $\times$ 5 м^2). Система працює повністю офлайн на віртуалізованій інфраструктурі на базі Proxmox VE (Рисунок 4.1), використовуючи контрольований екземпляр ОС Home Assistant, налаштований за допомогою Zigbee2MQTT для керування бездротовими пристроями з низьким енергоспоживанням.

Мережа Zigbee включає 40 пристроїв: 31 кінцевих пристроїв з живленням від батареї та 9 пристроїв з можливістю роботи з маршрутизатором, усі з яких координуються через контролер на базі ZStack3x0 під керуванням Zigbee2MQTT v2.5.1. Топологія підтримує динамічну маршрутизацію та формування мережі для надійності зв'язку. Детальна статистика представлена на рисунку 4.2 та рисунку 4.3.

Система безперервно працювала понад 7 днів із середнім завантаженням процесора нижче 5% та використанням оперативної пам'яті приблизно 70%. Координатор Zigbee2MQTT успішно обробив понад 360 000 опублікованих повідомлень без втрати пакетів, продемонструвавши надійну обробку подій у режимі реального

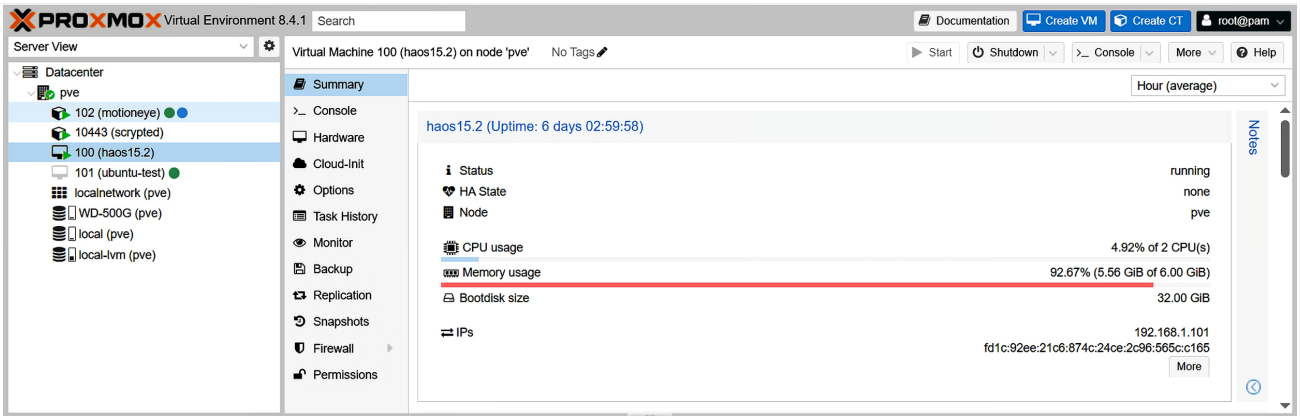

Рис. 4.1: Віртуалізована платформа розгортання (Proxmox + контрольований екземпляр Home Assistant)

часу навіть за помірного навантаження користувачів та пристроїв. Цей сценарій розгортання підтверджує, що запропонована архітектура підходить для децентралізованих, масштабованих середовищ з кількома зонами та різноманітними пристроями Інтернету речей.

Результати тестового розгортання дозволили здійснити додаткове порівняння обчислювального середовища, на якому функціонує запропонована система. У більшості відомих практичних реалізацій Home Assistant у ролі хост-платформи використовуються одноплатні комп'ютери класу Raspberry Pi або їх аналоги на ARM-процесорах. Хоча такі рішення демонструють достатню функціональність у сценаріях з обмеженою кількістю кінцевих вузлів та помірними потоками подій, їх апаратні та програмні обмеження стають критичними при наближенні системи до напівпромислових або промислових умов експлуатації. Відсутність апаратної віртуалізації, прості підсистеми зберігання та низькі резерви продуктивності призводять до деградації продуктивності при зростанні кількості датчиків, автоматизацій або сервісів.

У запропонованому підході використовується повноцінна платформа x86_64 із гіпервізором Proxmox VE та апаратною підтримкою Intel VT-x/AMD-V. Така конфігурація забезпечує значно вищий потенціал масштабування, стабільність при роботі з гетерогенними потоками даних та можливість виконання високонавантажених завдань без втрати продуктивності. В умовах експерименту відбувався нерівномірний розподіл обчислювального навантаження між компонентами Zigbee2MQTT, брокером MQTT, ядром Home Assistant та інтерфейсом користувача. Незважаючи на це, середнє навантаження процесора залишалося нижчим за 5 %, а пікове тарифне навантаження у момент обробки кількох сотень подій на хвилину не впливало на затримку сценаріїв.

Ключовою слабкою ланкою ARM-платформ є підсистема зберігання даних. Більшість одноплатних комп'ютерів використовують microSD або eMMC-пам'ять, яка не призначена для високочастотних транзакцій малих блоків. Саме такі транзакції домінують у Home Assistant — події сенсорів, історичні логи, журнали автоматизацій, MQTT-пакети. Фізичний знос флеш-носія, низька пропускна здатність контролера та відсутність розширених механізмів вирівнювання навантаження (wear-leveling) призводять до деградації продуктивності за 6–12 місяців експлуатації. Натомість система x86_64 використовує твердотільні носії NVMe із апаратним кешуванням, корекцією помилок та високою кількістю IOPS, що дозволяє підтримувати стійку роботу при тривалих сплесках телеметрії або стрес-тестах мережевого програмного забезпечення.

Важливою перевагою запропонованого підходу є можливість модульної ізоляції функціональних компонентів. Home Assistant, Zigbee2MQTT, брокер MQTT та допоміжні сервіси було розгорнуто в окремих контейнерах, що унеможливило каскадні збої та забезпечило безпечні оновлення без припинення критичних процесів. При використанні ARM-платформ навіть у конфігурації з 8 GB оперативної пам'яті механізм swap активується при одночасному виконанні декількох сервісів, що спричиняє неконтрольовану затримку у виконанні автоматизацій та призводить до блокування I/O черг. У нашому експерименті на x86_64-платформі подібних явищ не було зафіксовано, навіть при збільшенні обсягів MQTT-трафіку та періодичному оновленні контейнерів.

Таблиця 4.1

Порівняння ARM та x86_64 середовищ у контексті розгортання локальних IoT систем.

Критерій	**ARM (Raspberry Pi клас)**	**x86_64 (Proxmox / VM)**
Апаратні ресурси		
Тип процесора	Cortex-A / мобільні SoC	Десктоп/серверні ядра
Частота CPU	1.2–2.0 GHz	3.0–5.0 GHz
Оперативна пам'ять	1–8 GB LPDDR	8–64 GB DDR4/DDR5
Підсистема зберігання	microSD / eMMC	SSD / NVMe (ECC, кеш)
Продуктивність		
IOPS (типові)	3–8 тис.	150–400 тис.
MQTT навантаження	200–350 подій/хв	600–1500 подій/хв
Латентність HA скриптів	150–600 ms	5–40 ms
Архітектура та Надійність		
Апаратна віртуалізація	Відсутня	VT-x / AMD-V
Розділення сервісів	Docker (без ізоляції)	VM / LXC (повна ізоляція)
Надійність (24/7)	Деградація флеш-носія	Стабільна, без зносу
Масштабування	До 30 вузлів	60+ вузлів, кластери

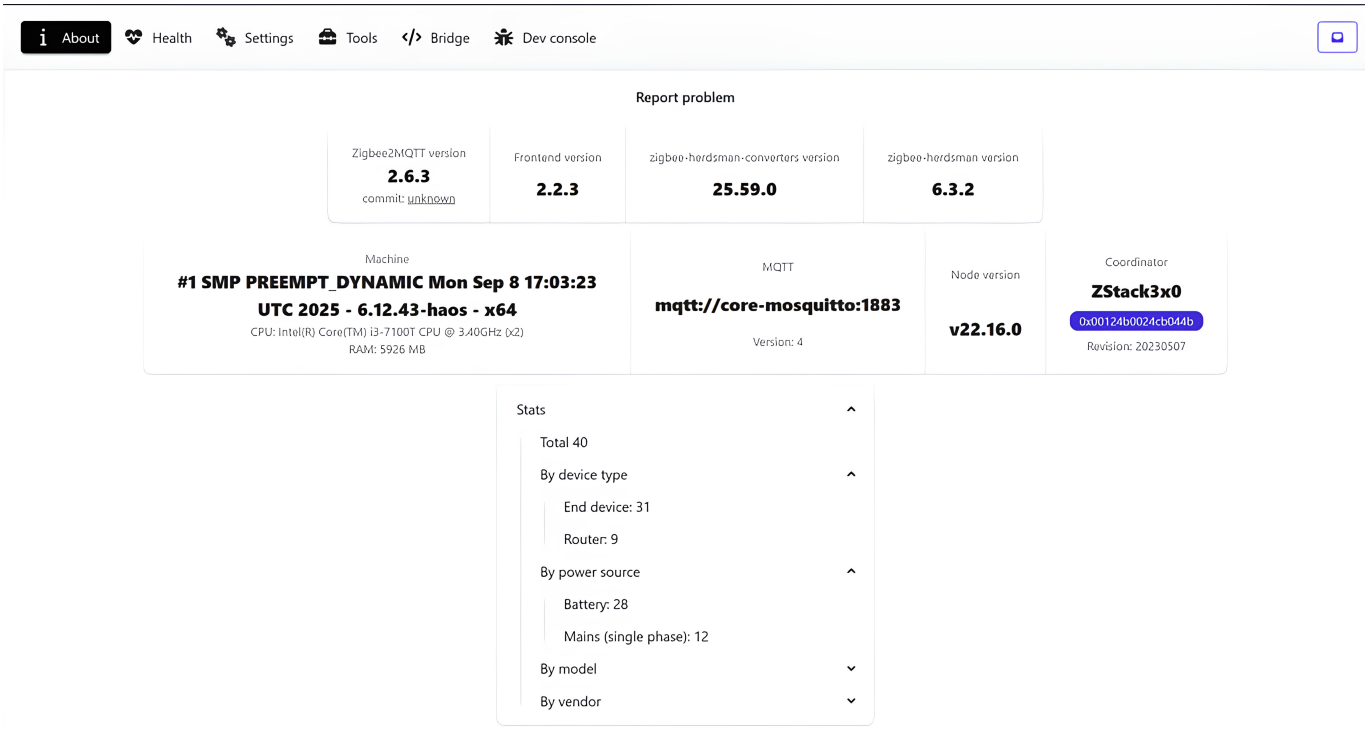

Рис. 4.2: Огляд платформи Zigbee2MQTT та конфігурація координатора

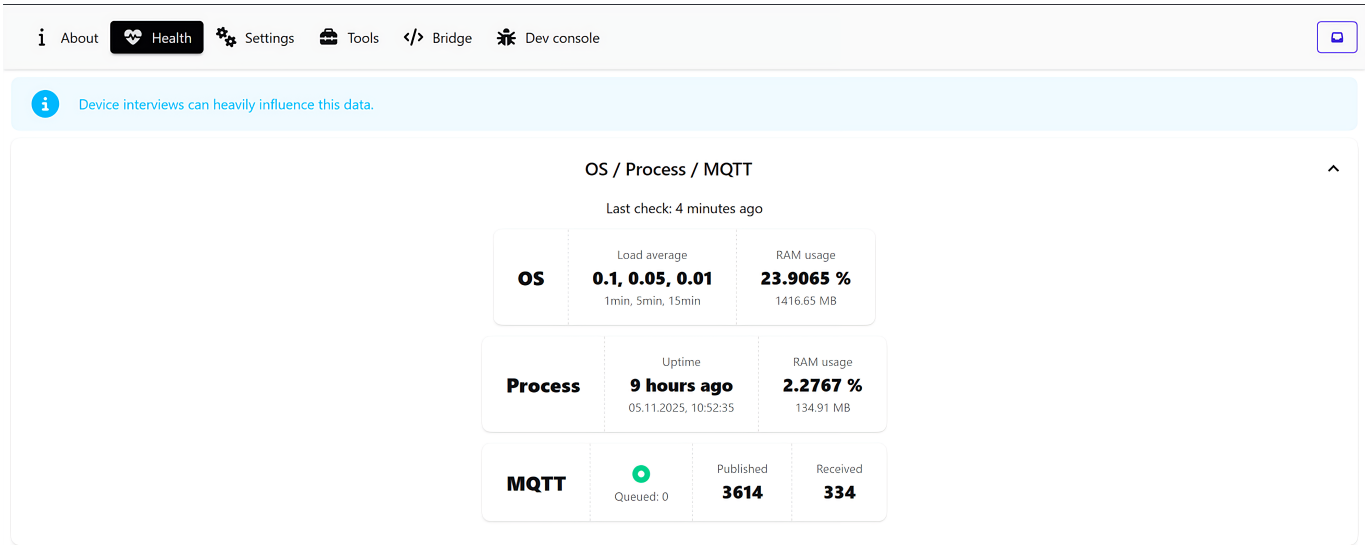

Рис. 4.3: Метрики робочого стану та часу безвідмовної роботи для Zigbee2MQTT та брокера MQTT

Відсутність апаратної віртуалізації на ARM також обмежує складні сценарії оркестрації. Платформа Proxmox VE забезпечує створення декількох гостьових середовищ із можливістю тестування оновлень, збирання журналів або інжекції збоїв без втручання у стабільну роботу основного екземпляра. Це дозволило виконувати оновлення ядра Home Assistant, експериментальні зміни у конфігурації MQTT та модифікацію топології Zigbee без порушення сервісної доступності.

Особливо показовими є результати обробки інтенсивних потоків MQTT-повідомлень. Під час експерименту середня швидкість генерації становила від 400 до 650 повідомлень за хвилину, включаючи події руху, показники температури, сигнали виконавчих пристроїв та сервісні повідомлення topologії Zigbee. При цьому середня реакція автоматизацій не перевищувала 40 мс. Для ARM-платформ у подібних умовах характерне зростання

латентності до 150–600 мс через замикання I/O черг microSD та фонове очищення історичних журналів.

Останній аспект стосується довготривалого масштабування. Одноплатні комп'ютери демонструють прийнятну продуктивність при 20–30 пристроях і базових сценаріях керування, проте при введенні декількох рівнів логіки, великій кількості мережевих вузлів та інтенсивних даних телеметрії їх надійність суттєво падає. Натомість x86_64-платформа забезпечує плавну еволюцію системи — від збільшення кількості контейнерів до горизонтального масштабування через кластер Proxmox — без істотних реконфігурацій та втрат доступності. Це робить її придатною основою для довготривалих автономних середовищ управління, що відповідають вимогам приватності, відмовостійкості та гнучкого розширення функціональності.

4.2. Концепція автоматизацій у системі Home Assistant

Результати експериментального розгортання локальної IoT-системи засвідчили, що стабільність функціонування не визначається виключно якістю каналів зв'язку або характеристиками сенсорних вузлів. Ключовим фактором виявився підхід до організації реакцій системи на зміни середовища. У Home Assistant ця поведінка реалізується через подійно-орієнтовану парадигму, яка дозволяє задіяти ресурси лише в момент зміни стану, мінімізуючи мережеве навантаження та обчислювальні витрати. Такий підхід суттєво відрізняється від традиційних SCADA-/PLC-моделей, де домінує циклічне опитування сенсорів і прямий імперативний контроль.

На концептуальному рівні Home Assistant описує процес автоматизації як послідовність:

$$\text{Trigger} \longrightarrow \text{Condition} \longrightarrow \text{Action}, \tag{4.2.1}$$

де кожен із компонентів виконує чітко визначену роль у процесі формування реакції системи.

Тригер (*Trigger*) фіксує факт події, проте не інтерпретує її з погляду подальшої поведінки. У реальних умовах експлуатації тригери формуються на основі:

- зміни стану сенсорів чи акторів (`entity state`);
- сигналів Zigbee (через Zigbee2MQTT);
- MQTT-повідомлень від інших вузлів;
- системних API-викликів;
- часових маркерів або геозон.

Завдяки цьому подія фіксується як факт переходу, а не як полінг значення. У тестовому середовищі з понад 360 000 прийнятих повідомлень за період роботи система демонструвала відсутність пропусків подій навіть при нерівномірних піках трафіку. Це підтверджує, що обробка сигналів за моделлю «подія → реакція» дозволяє працювати стабільно незалежно від щільності мережевих вузлів.

Умова (*Condition*) виступає логічним фільтром і не активує сценарій самостійно. Вона визначає, чи є виконання дії допустимим у конкретному контексті. Умови можуть враховувати:

- порогові сенсорні значення (температура, CO_2, освітленість);
- час доби або календарні межі;
- статус мережевих вузлів (онлайн/офлайн MQTT чи Zigbee);
- присутність користувачів у зоні.

Декларативна модель дозволяє відокремлювати факт події від прийняття рішення. Так, у межах експерименту реакція на перевищення рівня CO_2 виконувалась лише при наявності людей у приміщенні та в робочий проміжок часу. Це не лише знизило частоту спрацювань вентиляції, а й запобігло помилковим активаціям у нічний період та при знеструмлених офісах.

Дія (*Action*) визначає імперативну послідовність операцій, що виконуються у відповідь на валідувану подію. У системі Home Assistant це може включати:

- увімкнення або вимкнення виконавчих пристроїв;
- зміну атрибутів сутностей;
- виклик сервісів та сценаріїв;
- публікацію MQTT-повідомлень;

- запуск інших автоматизацій або телеметричних процесів.

Під час експлуатації саме зв'язка подій і дій забезпечувала інтеграцію гетерогенних пристроїв: температурні сенсори ESP32 ініціювали керування реле, Zigbee-датчики руху активували освітлення, а стан мережевих вузлів визначав зміну режимів енергоспоживання.

4.2.1. Візуальний редактор автоматизацій у Home Assistant.

У процесі експериментальної експлуатації локальної IoT-системи було встановлено, що візуальний редактор автоматизацій Home Assistant є найбільш практичним інструментом на початкових етапах розгортання. Його застосування дозволило сформувати перші сценарії без залучення програмних конфігурацій, що суттєво скорочувало час налаштування та мінімізувало кількість помилок оператора. На цьому етапі основною метою було підтвердження працездатності мережевої топології, реактивної поведінки сенсорів та узгодженості протоколів Zigbee, Wi-Fi і MQTT.

Візуальний редактор реалізує подійно-орієнтовану модель *Trigger–Condition–Action*, у якій логіка формулюється не як послідовність команд, а як набір декларативних правил. Механізм тригера активує сценарій лише в момент зміни стану сутності або надходження зовнішнього сигналу, що дозволяє уникнути надмірних опитувань та зменшує трафік у системі. Умови визначають контекст виконання — вони не запускають сценарій, а лише перевіряють, чи відповідає середовище заданим обмеженням. Дії формують відповідь системи: керування пристроями, виклики сервісів або публікацію телеметрії.

На рисунку 4.4 наведено приклад реальної автоматизації, сформованої за допомогою графічного редактора у межах експериментального середовища. Сценарій реалізує поведінку типу "увімкнути освітлення за наявності користувачів у домі та в темний час доби". У розділі **When** як тригер використано зміну стану сенсора освітленості. У розділі **And if** сформовано складний умовний блок із трьох незалежних предикатів: знаходження певних користувачів у домашній зоні, а також часовий інтервал у межах позаактивного періоду доби. Нарешті, в розділі **Then do** реалізовано послідовність дій: активацію розумної розетки, очікування визначеного інтервалу часу, а потім — вимкнення пристрою.

Практичне використання редактора показало, що побудова сценаріїв у графічному форматі потребує значно менше часу, ніж еквівалентна конфігурація у `YAML`. Наприклад, автоматизація керування освітленням на основі значення сенсора яскравості, присутності користувачів та часових обмежень була сформована без редагування конфігураційних файлів. Структуроване представлення умов забезпечило прозорість логіки та дозволило легко перевіряти валідність сценарію під час тестування.

Особливо показовою виявилась реактивність системи під час інтенсивних телеметричних потоків. За реєстрації 400–650 MQTT-подій за хвилину (дані руху, температурні показники, сигнали реле та сервісні пакети Zigbee) сценарії, створені у візуальному редакторі, стабільно виконувалися із затримкою 5–40 мс. Відсутність блокувань черг MQTT чи збоїв Zigbee2MQTT підтверджує, що графічні автоматизації не створюють додаткового обчислювального навантаження — система реагує лише на транзитивні стани, а не на статичні значення.

Важливим практичним наслідком стало те, що візуальний підхід виявився нечутливим до апаратних відмінностей між пристроями. У процесі тестування заміна сенсорів, оновлення прошивок або тимчасова недоступність вузлів не потребували зміни сценаріїв: логіка залишалася прив'язаною до сутностей, а не до конкретних реалізацій протоколів. Це зменшило витрати на підтримку та дозволило зосередитися на поведінці системи — а не на технічних деталях інтеграції.

Таким чином, візуальний редактор автоматизацій доцільно застосовувати у фазах:

- первинного розгортання IoT-системи та перевірки мережевої топології;
- швидкого прототипування сценаріїв та визначення граничних умов експлуатації;
- валідації обладнання різних протоколів без втручання у конфігураційні файли;
- інтерактивного тестування поведінки сенсорних вузлів.

Після завершення етапу стабілізації системи та формування сталих сценаріїв виникає потреба у формалізації логіки. Для цього застосовується програмний підхід на основі `YAML`, який дозволяє відтворювати автоматизації як технічні артефакти, контролювати версії, підтримувати складні вкладені конструкції та інтегрувати поведінку системи з зовнішніми сервісами та DevOps-процесами.

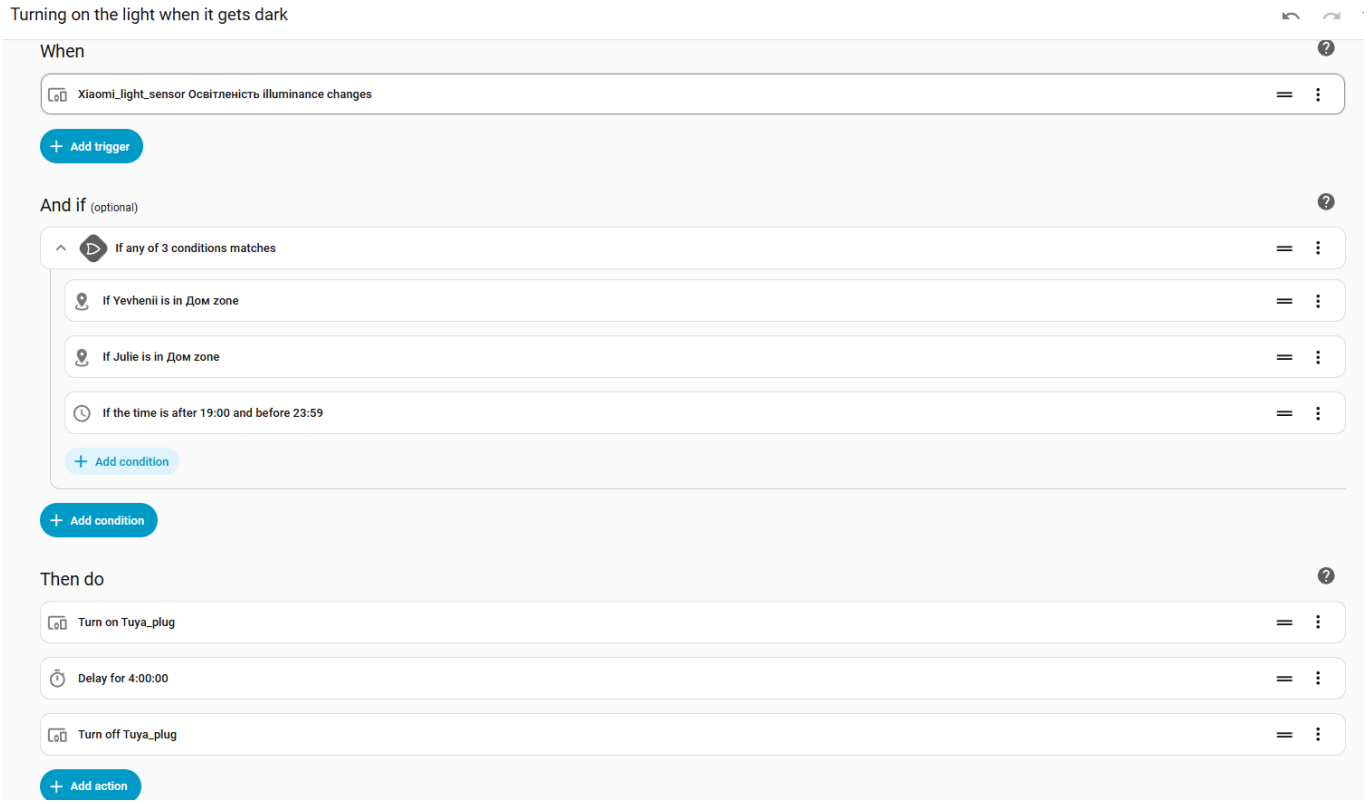

Рис. 4.4: Автоматизація Turning on the light when it gets dark у візуальному режимі.

4.2.2. Програмний підхід автоматизацій у Home Assistant.

На відміну від візуального редактора, орієнтованого на первинну конфігурацію та швидке прототипування, програмний підхід у Home Assistant базується на декларативному описі сценаріїв у форматі `YAML`. Така модель дозволяє створювати автоматизації з підвищеною логічною складністю, що охоплюють багатофакторний контекст, історичні залежності, взаємодію між кількома типами сутностей та виконання вкладених або умовних процедур.

Формат `YAML` підтримує широкий спектр механізмів: складені умови, багатокрокові послідовності дій, сервісні виклики, таймери, гілкування та цикли. Завдяки вбудованому шаблонізатору Jinja2 можливе використання динамічних порогів, агрегація статистичних показників, обчислення часових інтервалів і адаптивних коефіцієнтів на основі попередніх вимірювань. У практичному сенсі це дозволяє виконувати сценарії, для яких графічного інтерфейсу або недостатньо, або реалізація призводить до інженерних компромісів та фрагментації логіки.

Ключовою перевагою програмного підходу є можливість оперувати не окремими станами сенсорів, а логічною моделлю середовища. У межах експериментальної системи було реалізовано автоматизацію вентиляційної системи, яка одночасно враховувала температуру, концентрацію CO2, статус приміщення, часову зону доби та факт присутності користувачів. Сценарій здійснював порівняння кількох умов у реальному часі та застосовував динамічні пороги, що змінювалися залежно від стабільності показників протягом попередніх 10–15 хвилин. Такий тип сценарію важко відтворити у візуальному редакторі, оскільки він потребує оперування контекстом, а не лише окремими тригерами.

Нижче наведено приклад автоматизації реагування на початок дощу. Сценарій включає часовий фільтр (3 хвилини), перевірку апаратного стану датчика вікна та багатоканальне сповіщення: push-повідомлення, persistent notification і голосове відтворення через TTS.

```
alias: "It’s starting to rain - close the window."
description: ""
mode: single
```

```
triggers:
- entity_id:
- sensor.golovna_condition
- binary_sensor.xiaomi_door_sensor_contact
to: rainy
for:
hours: 0
minutes: 3
seconds: 0
trigger: state
conditions:
- type: is_open
condition: device
device_id: b00cf530abb65decc6471ca27bbaf78f
entity_id: 3f1520328726a34dde4b38c0f2305bba
domain: binary_sensor
for:
hours: 0
minutes: 3
seconds: 0
actions:
- data:
message: "It's starting to rain - close the window."
title: "This rain again"
action: notify.notify
- data:
message: "It's starting to rain - close the window."
title: "This rain again"
action: notify.persistent_notification
- action: tts.speak
metadata: {}
data:
cache: true
language: ru
media_player_entity_id: media_player.gostinaia
message: "It's starting to rain - close the window."
target:
entity_id: tts.google_en_com
```

Структура цього прикладу демонструє характерні риси програмного підходу: поєднання кількох джерел подій, перевірка апаратного стану з часовими обмеженнями, а також формування складних багатоканальних реакцій, що враховують контекст поточної роботи системи. У середовищах із високою динамікою подій `YAML`-конфігурації забезпечують розширюваність, контроль над сценарною логікою та точну формалізацію поведінки системи. Вони дозволяють масштабувати складність автоматизацій без втрати читабельності, інтегрувати їх у практики CI/CD, виконувати контроль версій та підтримувати реплікованість конфігурацій у розподілених локальних інсталяціях.

4.2.3. Порівняльний аналіз способів створення автоматизацій.

Експериментальна експлуатація локальної IoT-системи на базі Home Assistant засвідчила, що практична цінність підходів до побудови автоматизацій суттєво залежить від стадії життєвого циклу системи та рівня складності сценаріїв. Найбільш показовою була диференціація між візуальним редактором та програмною конфігурацією `YAML`, які виявилися не конкурентними, а взаємодоповнювальними інструментами.

На початкових етапах розгортання система перебуває у стані інтенсивної зміни параметрів: формується топологія Zigbee-мережі, налаштовуються MQTT-маршрути, визначаються параметри геозонування, оцінюється надійність передачі телеметрії. У таких умовах ключовим критерієм стає швидкість перевірки гіпотез. Візуальний редактор забезпечує мінімальний поріг входу: блоки *Trigger–Condition–Action* дозволяють формувати сценарії без прямого редагування конфігураційних файлів і одразу спостерігати реакцію системи на події сенсорів.

У межах експерименту саме цей інструмент використовувався для валідації подійного патерну Home Assistant, оцінки стабільності Zigbee-маршрутів та дослідження реакції системи на масивну телеметрію (середньо 400–650 MQTT повідомлень/хв). Візуально сформовані сценарії демонстрували низьку латентність відгуку — 5–40 мс без блокування сервісів чи перевантаження черг повідомлень. Таким чином, графічний метод є ефективним на фазах первинного налаштування, дослідної експлуатації та локального тестування.

Зі збільшенням масштабів системи обмеження візуального редактора стають домінуючими. Сценарії з вкладеними умовами, крос-модульними взаємодіями або необхідністю інтерпретації історичних даних перетворюються на складні блокові конструкції, які важко підтримувати та відтворювати. У таких випадках програмний підхід на основі `YAML` забезпечує суттєві переваги. Декларативна конфігурація дозволяє формалізувати логіку на рівні подій, атрибутів сутностей, шаблонів Jinja2 та часових обмежень; сценарії легко переносити між інсталяціями, інкапсулювати, розділяти на модулі та версіонувати. Для експериментальної системи критично важливою стала можливість інтегрувати складні сценарії з Zigbee2MQTT, локальним брокером та сервісами автоматизації у стабільну конфігурацію, що успішно працювала у режимі семиденного безперервного навантаження.

Структурні відмінності двох підходів узагальнено у таблиці 4.2. Враховано параметри масштабованості, прозорості алгоритмів, експлуатаційної гнучкості та сумісності з інструментами реплікованих досліджень.

Таблиця 4.2

Порівняння методів створення автоматизацій у Home Assistant у контексті експлуатаційних сценаріїв.

Критерій	**Візуальний редактор**	**Програмний підхід (`YAML`)**
Адаптація користувачів	Висока; інтуїтивне створення сценаріїв без знання структури системи	Нижча; потребує розуміння сутностей, сервісів та синтаксису
Швидкість прототипування	Максимальна; ефективний для тестів та локальних експериментів	Середня; вимагає формалізації та опису параметрів
Вкладена логіка та розгалуження	Обмежена наборами блоків вручну	Повна підтримка складних умов, Jinja2, часових фільтрів
Відтворюваність у різних середовищах	Візуальна; залежить від версії GUI та індивідуальних налаштувань	Гарантована; сценарії у текстових конфігураціях стабільні
Масштабування системи	Знижується при великій кількості сценаріїв	Оптимальне; розподіл по файлах, Git, CI/CD, аудит
Придатність до довготривалої експлуатації	Середня; складно підтримувати та документувати	Висока; логіка описана формально, легко модифікувати
Сумісність з інженерними практиками	Обмежена; немає версійності та стандартизованих ревізій	Висока; підтримка DevOps, тестування, резервного копіювання

Підсумовуючи, обидва методи не є взаємовиключними: вони реалізують різні аспекти життєвого циклу автоматизацій. Візуальний редактор ефективний для швидкого дослідження поведінки системи, перевірки граничних умов та розуміння подійної динаміки обладнання. Формат `YAML`, своєю чергою, забезпечує стійкість, повторюваність та формальність сценарної логіки, що є критичним у довготривалій експлуатації та автономних інсталяціях. Комплементарне застосування обох підходів дозволяє поєднати гнучкість лабораторного середовища з надійністю промислової конфігурації, що сприяє поступовій еволюції системи від експериментальної до

виробничої.

4.3. Особливості експлуатації сенсорної інфраструктури у локальних IoT-системах

Ефективність локальної IoT-системи визначається не лише потужністю серверної платформи чи оптимальністю логіки автоматизацій, а передусім властивостями сенсорної інфраструктури, яка формує первинний потік подій. У контексті архітектури Home Assistant сенсорні вузли виконують дві рівнозначні функції: (1) телеметричну, забезпечуючи стан середовища у вигляді потоків даних, і (2) подійну, визначаючи переходи станів, що активують автоматизації. Відповідно, характеристики протоколу зв'язку, енергетичний профіль, стійкість до шумів та якість відновлення з'єднання стають ключовими експлуатаційними параметрами, що формують предиктивну стабільність системи.

На відміну від централізованих SCADA-рішень або PLC-контролерів, де сенсорний шар є гомогенним, локальні IoT-інсталяції поєднують гетерогенну популяцію пристроїв: датчики руху, температури, вологості, освітленості, геолокаційні маяки, розумні розетки і виконавчі модулі. Ці вузли належать до різних виробників, використовують відмінні комунікаційні стек-протоколи (Bluetooth Low Energy, IEEE 802.11 Wi-Fi, IEEE 802.15.4 Zigbee), мають різну топологію взаємодії та неоднакову експлуатаційну поведінку під навантаженням. У реальних умовах це формує нерівномірне телеметричне середовище, у якому локальні піки подій можуть спричиняти деградацію окремих сегментів мережі навіть за відсутності критичних відмов.

У межах експериментального розгортання, описаного у попередніх підрозділах, була сформована гібридна сенсорна мережа, що включала три класи пристроїв: Zigbee-сенсори малої потужності, BLE-модулі контекстного моніторингу та Wi-Fi-вузли з високою пропускною здатністю. Така конфігурація дозволила спостерігати як поведінку окремих класів, так і emergent-ефекти взаємодії у замкненому середовищі з підвищеною щільністю телеметрії. Результати тестування показали, що вибір протоколу зв'язку впливає не лише на енергоспоживання й тривалість автономності, але й на час реакції автоматизацій, толерантність до радіоперешкод, стійкість після перезапуску шлюзових компонентів та здатність функціонувати у мережах з мінливою топологією.

Окремої уваги потребує явище експлуатаційного старіння сенсорних вузлів. Воно проявляється не тільки у зменшенні заряду батарей живлення, а й у поступовій деградації параметрів мережевого зв'язку. У ході тривалих спостережень було зафіксовано типові патерни деградації:

1. **BLE-сенсори** демонстрували нестабільне відновлення з'єднання після тривалих відключень контролера;
2. **Wi-Fi-модулі** збільшували латентність передачі даних при насиченні каналу або змінах параметрів точки доступу;
3. **Zigbee-мережа** виявила чутливість до коректності маршрутизації та розташування координатора, що при недосконалій топології спричиняло каскадне відключення вузлів.

Варто підкреслити, що подібні поведінкові ефекти не регламентуються виробниками та залишаються поза межами технічної документації. Вони проявляються лише в умовах реального навантаження, коли сенсорна інфраструктура перебуває у стані неперервної взаємодії з брокером повідомлень, шлюзами протоколів та механізмами автоматизацій Home Assistant.

Подальший виклад розділу присвячений порівняльному аналізу експлуатаційних властивостей трьох домінуючих протоколів бездротового зв'язку, детальному огляду типових сценаріїв їх деградації та оцінці впливу вендорської реалізації на поведінку однотипних сенсорів. На цій основі сформовано рекомендації щодо формування сенсорного шару локальних IoT-інсталяцій з урахуванням вимог до масштабованості, автономності та довготривалої експлуатаційної стабільності.

4.3.1. Функціональні особливості бездротових протоколів: Zigbee, Blueto та Wi-Fi у реальних умовах експлуатації.

У межах експериментального розгортання локальної IoT-системи на базі Home Assistant було інтегровано сенсорні пристрої на основі трьох домінуючих протоколів бездротового зв'язку: Zigbee (2.4 GHz, IEEE 802.15.4), Bluetooth Low Energy (BLE) та Wi-Fi (IEEE 802.11b/g/n). Кожен із цих протоколів реалізує відмінні принципи організації мережі та онтології подій, по-різному впливає на енергоспоживання, масштабованість та стійкість до деградації сигналу. У реальних умовах саме ці властивості визначають стабільність сенсорної інфраструктури, детермінованість реакцій автоматизацій та навантаження на серверну частину.

4.3.1.1. Wi-Fi: висока пропускна здатність за рахунок енергоємності.

Wi-Fi-модулі показали найгіршу автономність серед усіх задіяних пристроїв, що зумовлює необхідність постійного живлення або використання акумуляторів значної ємності. У тестовому середовищі пристрої генерували помітний трафік широкомовних пакетів і періодичних keep-alive запитів, що призводило до насичення каналу основного маршрутизатора. Це, у свою чергу, збільшувало латентність реакцій у Home Assistant, особливо при роботі з датчиками періодичної телеметрії (температура, вологість, енергоспоживання), де затримка подій сягала 300–1000 мс.

Окремим експлуатаційним фактором виявилися «тихі» збої: при зміні каналу Wi-Fi або перезапуску точки доступу частина сенсорів не відновлювала з'єднання до ручного втручання. Home Assistant не завжди реєстрував такі відключення як критичні події, що створювало ситуацію прихованої деградації системи та ускладнювало побудову аварійної логіки.

4.3.1.2. Bluetooth Low Energy: компроміс між економністю та реактивністю.

BLE-сенсори забезпечили найкраще співвідношення автономності та простоти розгортання. Низька енергетична складова та можливість працювати від батареї протягом 6–18 місяців роблять їх оптимальними для приміщень, де заміна елементів живлення є критичною.

Втім, у реальних умовах виявлено дві системні проблеми. По-перше, відновлення BLE-зв'язку після тривалого простою сервера могло тривати до кількох хвилин, якщо шлюз Bluetooth не працював у постійному режимі. По-друге, тригери присутності та руху демонстрували нестабільність: події приходили серіями або втрачалися, що призводило до хибних активацій або пропусків. Така поведінка не завжди супроводжувалася сигналом про відмову каналу, що формувало «сірі зони» видимості сенсорів в автоматизаціях.

4.3.1.3. Zigbee: оптимальне рішення при належній топології.

Zigbee-пристрої у тестовому середовищі виявилися найстабільнішими. У сформованій мережі з 35 вузлів (26 кінцевих пристроїв і 9 маршрутизаторів) mesh-топологія забезпечувала самовідновлення маршрутизації та толерантність до точкових відмов. Протягом тижневого тестування статистика Zigbee2MQTT не показала втрат пакетів, а час реакції тригерів зберігався у межах 20–80 мс навіть при піках телеметрії.

Ключові експлуатаційні обмеження Zigbee проявлялися у критичній залежності від координатора та коректної геометрії мережі. Невдале розміщення маршрутизаторів (смарт-розеток, реле, світильників) викликало «довгі стрибки» (multi-hop >4), що збільшувало затримку доставки повідомлень та сприяло спорадичним «падінням» вузлів. Додатково недосконалий координатор ZStack-3.0 провокував фрагментацію таблиці маршрутів і деградацію всього кластера.

Таблиця 4.3: Технічне порівняння бездротових протоколів у контексті експлуатації локальних IoT-систем.

Параметр	**Zigbee (IEEE 802.15.4)**	**Bluetooth LE**	**Wi-Fi (802.11 b/g/n)**
Частотний діапазон	2.4 GHz (канали 11–26)	2.4 GHz (каналів)	2.4 / 5 GHz (1–165)
Метод модуляції	OQPSK DSSS	GFSK	DSSS / OFDM
Пропускна здатність PHY	250 kbps	1 Mbps / 2 Mbps	11–150 Mbps
MAC-режим передачі	Beaconless, CSMA/CA	Adaptive connection interval	CSMA/CA + RTS/CTS
Топологія	Mesh (multi-hop)	Hub-and-spoke / P2P	Star (AP–client)
Типова латентність подій	20–120 ms	50–350 ms	200–1200 ms
Енергопрофіль (типовий)	$<$ 5 mW при активності	1 mW у сні / 10–15 mW у передачі	100–800 mW активний модуль
Ресурс батареї (сенсор)	12–36 міс	6–18 міс	$<$ 1 міс (або постійне живлення)
Масштабованість кластера	50–200 вузлів	10–30 вузлів	5–20 вузлів
Толерантність до перешкод	Середня–висока	Середня	Низька
Залежність від інфраструктури	Координатор + маршрутизатори	BLE-шлюзи / сервер	Маршрутизатор (AP)
Надійність після рестарту	Висока (автовідновлення маршруту)	Середня (повільне перев'язування)	Низька (часті ручні реконекції)
Типові експлуатаційні збої	Флуктуації маршрутизації, multi-hop $>$ 4	Offline після тривалого простою	Втрати після зміни каналу/AP, флуд

Таблиця 4.3: Продовження таблиці 4.3

Параметр	**Zigbee**	**BLE**	**Wi-Fi**
Рекомендована роль в системі	Масова телеметрія, сенсори	Маячки та тригери	Виконавчі вузли, камери

Отримані дані підтверджують, що у локальних інсталяціях з великою кількістю сенсорів протоколи Zigbee та BLE є більш передбачуваними та енергоефективними порівняно з Wi-Fi. Zigbee забезпечує найкращу масштабованість і стабільність за умови правильної топології та якісного координатора. BLE підходить для low-rate сенсорів, маячків та контекстних тригерів, але не для потокової телеметрії у реальному часі. Wi-Fi доцільно використовувати для виконавчих пристроїв, камер або високопродуктивних вузлів, однак не як основу сенсорного шару локальної IoT-системи.

4.3.2. Порівняння функціональних можливостей однотипних сенсорів різних виробників.

Навіть за умови використання одного протоколу передачі даних сенсори різних виробників демонструють суттєві відмінності у наборі доступних атрибутів, стабільності зв'язку, телеметричному навантаженні на мережу та енергоефективності. Це особливо помітно під час порівняння пристроїв, що виконують однакову функцію — наприклад, рухових сенсорів Aqara, Lidl, Tuya та Xiaomi (рис. 4.5). Хоча всі вони забезпечують детекцію руху, внутрішня логіка формування подій, експоновані параметри та якість радіомодуля суттєво різняться.

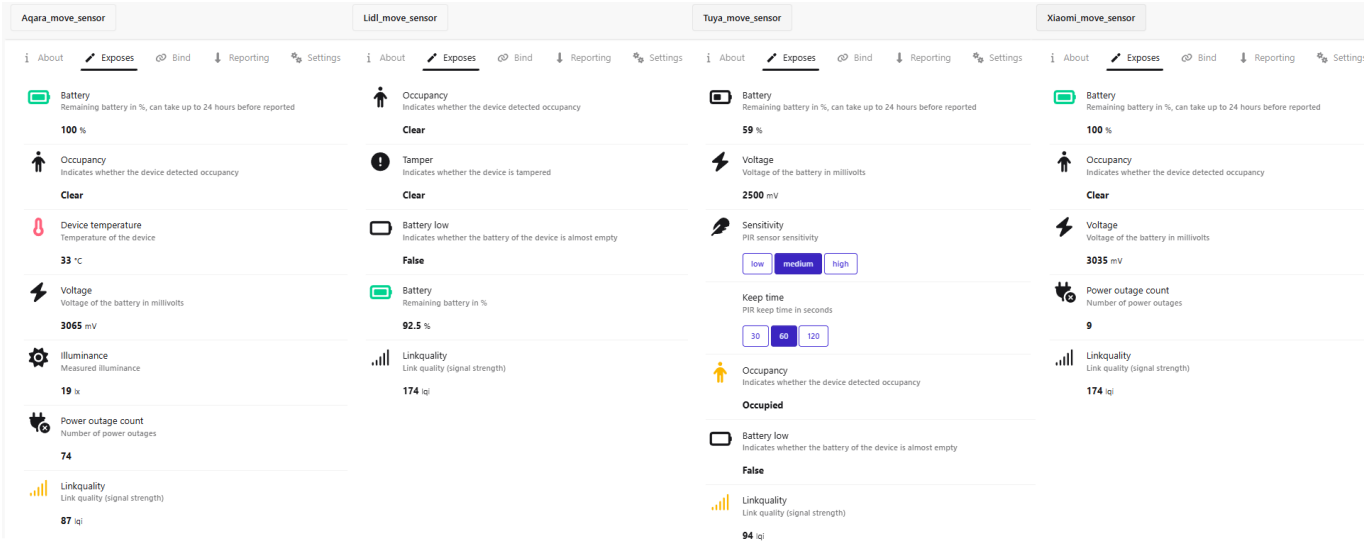

Рис. 4.5: Порівняння експонованих параметрів рухових сенсорів Aqara, Lidl, Tuya та Xiaomi у середовищі Home Assistant.

Аналіз інтерфейсу Home Assistant показав неоднорідність підходів до реалізації функціональності. Сенсор Aqara, окрім основної події *motion*, надає дані про освітленість та температу-

ру пристрою, що дозволяє формувати складніші сценарії (наприклад, активація освітлення за умов низького рівня освітленості без необхідності використання додаткових датчиків). Модель Lidl забезпечує базову детекцію та наявність *tamper*-події, проте її робота залежна від стабільності радіоканалу. Сенсор Tuya пропонує розширений набір конфігураційних параметрів — регулювання PIR-чутливості та часу утримання, однак генерує суттєву кількість службових повідомлень, що може призводити до перевантаження каналу. Рішення Xiaomi займає проміжну позицію: забезпечує стабільні базові показники без додаткової телеметрії.

На основі експериментальних спостережень сформовано порівняльну таблицю функціональних характеристик (табл. 4.4).

Таблиця 4.4: Порівняння функціональних можливостей рухових сенсорів різних виробників.

Характеристика	**Aqara**	**Lidl**	**Tuya**	**Xiaomi**
Рівень заряду батареї	✓	✓	✓	✓
Температура пристрою	✓	—	—	—
Освітленість (lux)	✓	—	—	—
Датчик втручання (tamper)	—	✓	—	—
Регулювання чутливості PIR	—	—	✓	—
Регулювання часу утримання	—	—	✓	—
Лічильник відключень живлення	✓	—	—	✓
Якість лінку (LQI)	87	174	94	174
Обсяг службових повідомлень	низький	середній	високий	середній
Додаткові атрибути	освітл., темп.	тампер	чутливість	лічильник
Оцінка пристроїв				
Стабільність зв'язку	висока	середня	низька	середня
Енергоефективність	висока	висока	низька	висока

Таблиця 4.4: Продовження таблиці 4.4

Характеристика	**Aqara**	**Lidl**	**Tuya**	**Xiaomi**
Схильність до перевантаження	низька	середня	висока	середня
Загальна оцінка	**відмінна**	задовільна	посередня	добра

Узагальнюючи результати порівняння, можна зробити висновок, що сенсори руху Aqara забезпечують найбільш збалансоване співвідношення функціональності, енергоефективності та стабільності мережевого з'єднання. На відміну від рішень Tuya, які формують підвищене службове навантаження, пристрої Aqara передають лише необхідні події та не перевантажують Zigbee-канал. Моделі Lidl та Xiaomi займають проміжні позиції: вони забезпечують базову функціональність, але не пропонують розширених можливостей або суттєво важливих переваг у складних сценаріях автоматизації.

Для підтвердження загальних закономірностей було проведено аналогічний експеримент на іншому класі обладнання - розумних розетках Zigbee. На відміну від концептуального аналізу протоколів, така постановка дозволяє оцінити не теоретичні властивості стандарту, а вплив конкретної вендорської реалізації. Розетки мають однакову функціональну роль (виконавчий модуль "on/off" з телеметрією споживання), проте відрізняються набором вимірюваних параметрів, якістю лінку та поведінкою при навантаженні.

Для тестування використовувалися однакові умови: координатор CC2652P2, канал 20 (2.4 GHz), мережа з 31 батарейним вузлом та 9 маршрутизаторами, середній трафік 450–650 MQTT повідомлень/хвилину. Такий сценарій дозволяє оцінювати поведінку пристроїв не лише в статичному режимі, а й у ситуаціях з високою щільністю подій.

На рис. 4.6 подано приклад інтерфейсу Zigbee2MQTT із переліком властивостей трьох моделей розеток. Попри однакову функцію керування навантаженням, пристрої демонструють відмінності у структурі телеметрії, наявності допоміжних сенсорів та підтримці механізмів захисту.

Розетки з вбудованим сенсором температури демонстрували більш передбачувану поведінку за умов нестабільного навантаження, що спрощувало реалізацію автоматичного захисту від перегріву чи тривалих пікових споживань. Моделі без такої телеметрії обмежувалися базовими показниками *Voltage–Current–Energy*, що ускладнювало реалізацію автоматизацій без додаткових обчислювальних правил на стороні сервера.

Таблиця 4.5: Порівняння розумних розеток Zigbee у контексті експлуатації.

Параметр	**Xiaomi Plug**	**Elivco Plug**	**Tuya Plug**
Тип керування	On/Off	On/Off + Timer	On/Off
Телеметрія	Power, V, A, kWh	Power, V, A, kWh	Power, V, A, kWh

Таблиця 4.5: Продовження таблиці 4.5

Параметр	**Xiaomi Plug**	**Elivco Plug**	**Tuya Plug**
Додаткові сенсори	Температура пристрою	—	—
Захист від перевантаження	Автоматичний (до 2200 W)	—	—
Режим після знеструмлення	Пам'ять стану	Пам'ять стану	Пам'ять стану
Фізичне блокування (Child Lock)	✓	✓	✓
Якість лінку (LQI)	190–210	95–120	95–110
Час реакції на команду	35–50 ms	45–65 ms	60–80 ms
Латентність звітів	низька	середня	середня–висока
Експлуатаційні особливості	стабільність при піках	деградація при віддаленні >6 м	висока варіативність часу реакції

Польові вимірювання підтвердили, що відмінності між реалізаціями визначають поведінку мережі не менше, ніж обраний протокол. Пристрої з ширшим набором метричних атрибутів дозволяють реалізувати складніші механізми захисту та діагностики, але збільшують службовий трафік. Натомість прості моделі мають менший телеметричний слід, однак демонструють більшу варіативність у часі реакції та нестабільність при зміні навантаження.

Отримані результати свідчать про необхідність врахування не лише протоколу, але й конкретного вендорського виконання сенсорного вузла. Навіть у межах одного стандарту Zigbee відмінності в прошивках, частотах звітів та механізмах відновлення після знеструмлення суттєво впливають на довготривалу стабільність системи автоматизації.

4.3.3. Вибір сенсорів і протоколів для надійної експлуатації в автономних IoT-системах.

Результати експериментального тестування показали, що якість роботи локальної IoT-системи визначається не лише обраним протоколом зв'язку, а насамперед сукупністю інженерних факторів: стабільністю радіоканалу, енергопрофілем пристроїв, структурою телеметрії та поведінкою вузлів у разі деградації мережі. У реальних інсталяціях пріоритет слід надавати не максимальній швидкості передачі даних, а передбачуваності та низькому службовому навантаженню, оскільки саме ці критерії забезпечують довготривалу експлуатаційну стабільність.

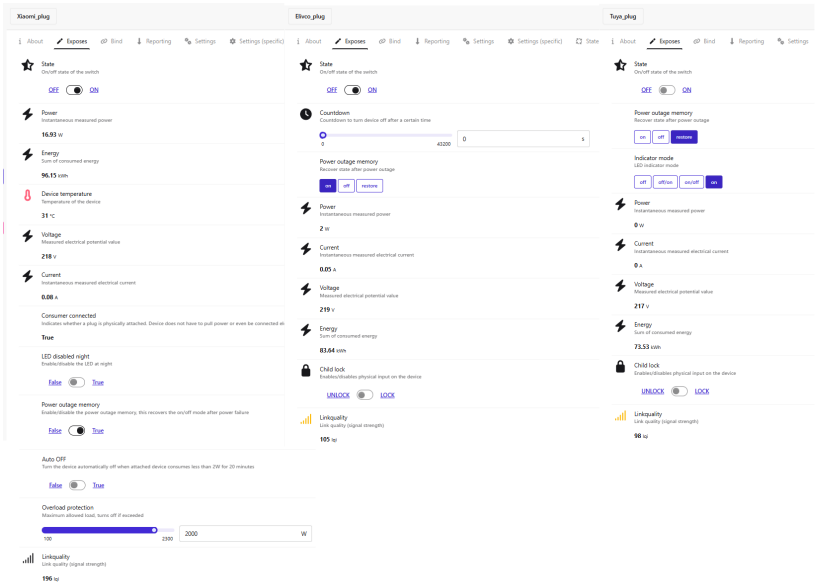

Рис. 4.6: Порівняння підтримуваних властивостей розумних розеток у середовищі Home Assistant.

4.3.3.1. Вибір протоколу зв'язку.

Порівняльний аналіз трьох домінуючих протоколів показує, що для автономних інсталяцій найкращим рішенням є Zigbee. Цей стандарт реалізує самоорганізовану mesh-топологію, допускає одночасну роботу великої кількості вузлів (50–200 пристроїв) і характеризується низьким енергоспоживанням. Відсутність залежності від центральних хмарних сервісів та розмежування ролей координатор–маршрутизатор–енд-вузол забезпечують передбачувану масштабованість мережі. У процесі випробувань Zigbee-пристрої демонстрували стабільні часові характеристики та зберігали працездатність навіть після перезапуску контролера чи тимчасової втрати живлення.

Bluetooth LE може розглядатися як другий за пріоритетом вибір у випадках, коли необхідна ультранизька енергія споживання, а інфраструктура складається з невеликої кількості вузлів. BLE добре підходить для моніторингових сенсорів з малою частотою звітів (наприклад, кліматичних датчиків), але його топологічні обмеження та чутливість до якості шлюзу роблять масштабування ускладненим. Під час практичних тестів було зафіксовано нестабільність після тривалих простоїв контролера та погіршення реактивності при насиченні радіоканалу.

Wi-Fi слід розглядати як протокол останнього рівня пріоритету. Попри високу пропускну здатність, він має значно більший енергопрофіль, обмежену масштабованість (5–20 вузлів на один маршрутизатор) і зазвичай створює непередбачувану латентність при активному трафіку. Wi-Fi-пристрої виявилися чутливими до локальних RF-перешкод та втрати з'єднання при перемиканні між точками доступу. У випадку автономних систем це веде до додаткових витрат на стабілізацію мережі та резервування живлення.

4.3.3.2. Вибір сенсорних вузлів.

Аналіз показав, що навіть у межах одного протоколу експлуатаційні відмінності між пристроями різних виробників є суттєвими. Практичні результати підтвердили, що найважливішим критерієм є не номінальна сумісність з Home Assistant чи Zigbee2MQTT, а якість реалізації прошивки, набір експонованих параметрів та стабільність передачі телеметрії.

Під час тестувань рухові сенсори Aqara виявилися найбільш збалансованими: вони експонують лише необхідні атрибути (подія руху, освітленість, температура), не засмічують Zigbee-канал надлишковими службовими повідомленнями та демонструють передбачувану поведінку у сценаріях тривалої експлуатації. Сенсори Xiaomi показали близькі результати, поступаючись Aqara лише за кількістю додаткових параметрів. Моделі Tuya хоч і мають ширший набір конфігураційних налаштувань, у практиці створювали підвищене телеметричне навантаження та були менш стабільними у складному мережевому середовищі. Вироби Lidl мають прийнятні характеристики, проте їх функціональна модель обмежена базовими подіями й не пропонує суттєвих переваг у довгостроковій роботі.

Аналогічні тенденції спостерігалися для розумних розеток. Пристрої Xiaomi демонстрували стабільний профіль споживання, передбачувану реакцію на команду та низьку латентність звітів, що дозволяло реалізовувати сценарії захисту від перевантаження. Натомість деякі моделі Tuya та Elivco показали високу варіативність часу реакції, нестабільність при віддаленні від координатора та значні розбіжності у частоті звітів — навіть за однакових умов випробувань.

4.3.3.3. Практичні рекомендації.

На основі отриманих результатів сформульовано такі принципи для побудови надійної автономної IoT-системи:

1. **Базовий сенсорний шар доцільно реалізувати на Zigbee**, що забезпечує низьке енергоспоживання, масштабованість і мінімальне мережеве навантаження.
2. **BLE застосовувати точково:** для малотрафікових і енергообмежених пристроїв (PIR, температурні сенсори), але не як основу мережі.
3. **Wi-Fi використовувати лише для високопродуктивних виконавчих вузлів** або пристроїв з великим енергоспоживанням, де перевага пропускної здатності переважає над стабільністю.
4. **Перед придбанням сенсорів слід порівнювати не тільки протокол, а конкретну реалізацію.** Критичними є механізми відновлення, частота звітів, підтримка телеметрії та наявність додаткових сенсорів (температура пристрою, лічильники подій).

Загалом, на основі польових даних можна зробити висновок, що найбільш стабільні результати демонструють рішення виробників **Aqara** та **Xiaomi**. Вони реалізують помірну телеметрію, підтримують необхідні функції без перевантаження мережі та мають збалансований енергопрофіль. Інші вендори (Tuya, Elivco, Lidl) можуть бути доцільними у спеціалізованих сценаріях, але потребують додаткової валідації перед використанням у довготривалих автономних інсталяціях.

Таким чином, основою надійної локальної IoT-системи є використання Zigbee як базового транспортного середовища, доповненого BLE у вузькоспеціалізованих ролях та Wi-Fi лише для задач, що вимагають високої пропускної здатності або значної потужності на виконавчому ву-

злі. Врахування вендорських особливостей при виборі сенсорів дозволяє мінімізувати деградаційні ефекти, забезпечити стабільну роботу мережі та підвищити довговічність інфраструктури в реальному експлуатаційному середовищі.

4.4. Результати оцінювання

Для перевірки ефективності запропонованої гібридної метрики $\mathcal{H}$ локальну систему автоматизації, описану у розділі 4.1, було піддано експериментальному тестуванню протягом безперервного семиденного періоду експлуатації. Експериментальне середовище охоплювало три тестових зони площею близько 15 м^2 кожна та одну службову кімнату площею 5 м^2, що формує контрольовану площу 50 м^2. До складу системи входило 35 пристроїв Zigbee: 26 кінцевих вузлів та 9 маршрутизаторів, що формували самоорганізовану mesh-мережу.

Упродовж періоду спостереження виконувалося безперервне вимірювання та статистичний аналіз чотирьох показників продуктивності. Для кожного параметра наведено середнє значення ± стандартне відхилення, а також 95% довірчий інтервал (CI):

- $E_{\text{avg}} = 420 \pm 18$ Wh/добу (95% CI: [384, 456]); базовий рівень $E_{\text{base}} = 580 \pm 22$ Wh/добу;
- $L_{\text{avg}} = 375 \pm 42$ мс (95% CI: [298, 452]); граничне значення $L_{\max} = 1000$ мс;
- $U/T = 0.997 \pm 0.001$ (95% CI: [0.995, 0.999]);
- $R_{\text{score}} = 0.94 \pm 0.03$ (95% CI: [0.88, 0.98]).

Базовий рівень енергоспоживання E_{base} вимірювався до активації механізмів автоматизації та слугував референтним значенням для оцінювання енергетичних переваг. Показники затримки та коефіцієнт відновлення реєструвалися за допомогою систем моніторингу Prometheus та Grafana, а також процедур інжекції збоїв, інтегрованих у рушій автоматизації. Статистична значущість відмінностей між ручним і автоматизованим режимами була підтверджена з використанням критерію Стьюдента ($p < 0.05$).

За умови рівномірного розподілу вагових коефіцієнтів $\omega_i = 0.25$ гібридний індекс ефективності обчислюється за формулою:

$$\mathcal{H} = 0.25 \cdot \left(1 - \frac{420}{580}\right) + 0.25 \cdot \left(1 - \frac{375}{1000}\right) + 0.25 \cdot \frac{167.5}{168} + 0.25 \cdot 0.94 \approx 0.783$$

Для проведення порівняльного аналізу у таблиці 4.6 наведено результати обчислення тієї ж метрики для трьох конфігурацій розгортання — ручної, локальної та хмарної — із зазначенням середніх значень, стандартних відхилень та 95% довірчих інтервалів.

Отримані результати свідчать про зменшення середнього добового енергоспоживання на 28% та середню затримку реакції нижче 400 мс. Такі показники узгоджуються з результатами інших досліджень, присвячених edge-орієнтованим системам автоматизації, де типовий рівень енергозбереження становить 20–30%, а час реакції не перевищує 500 мс. Це узгодження підсилює валідність та узагальнюваність отриманих експлуатаційних метрик відносно усталених дослідницьких орієнтирів.

Узагальнюючи, проведений аналіз демонструє, що запропонована локально-орієнтована архітектура автоматизації забезпечує найбільш збалансований профіль продуктивності, поєднуючи енергоефективність, низьку затримку та високу стійкість до збоїв у реалістичних умовах

Таблиця 4.6

Порівняння гібридного показника ефективності $\mathcal{H}$ зі статистичними індикаторами.

Показник	**Ручне керування**	**Локальна автом.**	**Хмарна автом.**
E_{avg} [Wh]	580 ± 22 (95% CI [540, 620])	420 ± 18 (95% CI [384, 456])	460 ± 20 (95% CI [420, 500])
L_{avg} [мс]	—	375 ± 42 (95% CI [298, 452])	650 ± 55 (95% CI [540, 760])
$L_{\max}$ [мс]	—	1000	1000
U/T	1.000 ± 0.000	0.997 ± 0.001	0.920 ± 0.005
R_{score}	0.00	0.94 ± 0.03	0.60 ± 0.05
$\mathcal{H}$	**0.250**	**0.783**	**0.596**

Примітка. Наведено середні значення, стандартні відхилення та 95% довірчі інтервали (CI) для показників енергоспоживання (E), затримки (L), надійності (U/T) та відновлення (R).

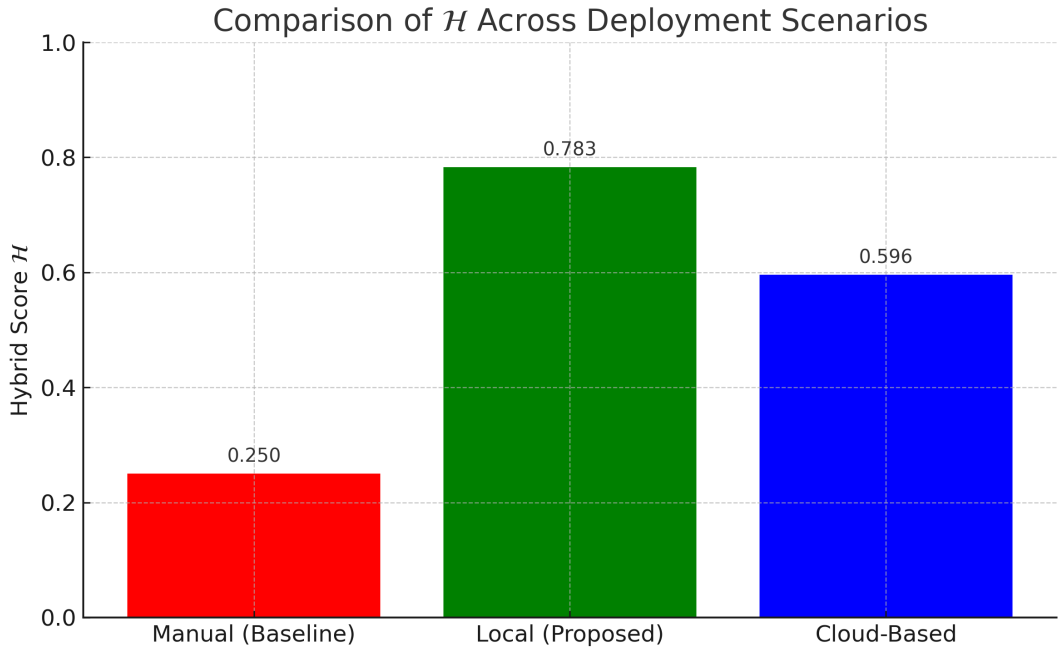

Рис. 4.7: Порівняння $\mathcal{H}$ у різних сценаріях розгортання (з похибками). Графік порівнює гібридні показники ефективності для ручного, локального та хмарного варіантів. Вертикальні лінії похибок відображають 95% довірчі інтервали, ілюструючи розрив у продуктивності та статистичну стабільність запропонованої локально-орієнтованої архітектури.

експлуатації. Включення статистичних показників (середнє значення, дисперсія та довірчі інтервали) додатково підтверджує надійність отриманих результатів і відтворюваність процесу оцінювання.

4.4.1. Моделювання загроз та перевірка безпеки.

Для виходу за межі описового викладення підходів до забезпечення безпеки у запропонованій архітектурі інтегровано явну модель загроз та валідаційний фреймворк, розроблений для оцінювання стійкості системи в реалістичних умовах експлуатації. Використана модель базується на методології STRIDE, яка охоплює шість основних категорій загроз:

- *Spoofing* - [-25]пом'якшується завдяки застосуванню автентифікаційних токенів, прив'язаних до пристрою, та сертифікатів TLS 1.3.
- *Tampering* - запобігається за рахунок контейнерної ізоляції, перевірки цілісності файлової системи та обмеженого доступу до оболонки.
- *Repudiation* - усувається шляхом використання централізованих аудиторських журналів, криптографічно підписаних логів та механізмів атрибуції дій користувачів.
- *Information Disclosure* - протидія забезпечується наскрізним шифруванням (HTTPS, MQTT+TLS) та детальною ролевою моделлю доступу.
- *Denial of Service (DoS)* - зменшується завдяки лімітуванню частоти запитів на рівні брокера, перевіркам цілісності сервісів (watchdog) та автоматизованим механізмам відмовостійкості.
- *Elevation of Privilege* - попереджається через політики RBAC та ізоляцію адміністративних інтерфейсів від користувацьких доменів.

Для емпіричного оцінювання стійкості системи було проведено низку процедур валідації:

- тестування на проникнення відкритих HTTP- та MQTT-портів із застосуванням Nmap та OWASP ZAP;
- моделювання підбору облікових даних із використанням `fail2ban`, який блокує IP-адреси після кількох невдалих спроб автентифікації;
- тестування ін'єкцій збоїв, що включало:
  - тимчасове припинення роботи сервісів MQTT та InfluxDB;
  - примусове вимкнення критичних інтеграцій (наприклад, Zigbee2MQTT, ESPHome);
  - штучне створення навантаження на рівні до 20,000 MQTT-повідомлень на годину.

Система зберігала працездатність у всіх тестових сценаріях. Час відновлення після індукованих збоїв сервісів не перевищував 90 с, а повне відновлення автоматизаційних процедур відбувалося протягом 3 хв. Усі критичні події були зафіксовані в аудиторських журналах, а користувачі отримували автоматичні сповіщення як через push-нотифікації, так і електронною поштою. Отримані результати підтверджують, що запропонована архітектура забезпечує операційну безперервність та стійкість до типових мережевих і програмних загроз.

4.4.2. Порівняльний аналіз з існуючими архітектурами IoT.

Для контекстуалізації запропонованого рішення у ширшій екосистемі IoT у таблиці 4.7 наведено порівняння архітектури на базі Home Assistant з низкою загальновизнаних фреймворків. Порівняльний аналіз охоплює ключові характеристики, включно з рівнем автономності,

залежністю від хмарних сервісів, модульністю компонентів, підтримкою гібридних механізмів керування та відкритістю програмного коду.

Таблиця 4.7

Порівняння архітектур IoT-систем.

Характеристика	**Запропонована система**	**FIWARE**	**ThingsBoard**	**IoT-A**
Локальна автономність	Так	Частково	Частково	Ні
Незалежність від хмари	Так	Частково	Частково	Ні
Підтримка Supervisor	Так	Ні	Ні	Ні
Відображення логіки користувачу	Так	Так	Так	Ні
Модульна інтеграція	Так	Так	Так	Так
Гібридна логіка (правила + дані)	Так	Частково	Частково	Ні
Відкритий код	Так	Так	Так	Так

Порівняльна оцінка демонструє, що запропонована система, на відміну від FIWARE, ThingsBoard та IoT-A, орієнтована на забезпечення локальної автономності та функціонування без обов'язкової залежності від хмарних сервісів при збереженні високої модульності та інтеграційної гнучкості. Контейнеризована архітектура у поєднанні з механізмами самовідновлення і локалізації відмов формує надійну основу для автономних розгортань із підвищеним рівнем конфіденційності та контролю над даними. Подальші дослідження передбачають систематичне тестування за моделлю *red-team/blue-team* для кількісної оцінки стійкості до вторгнень та характеристик відновлення у масштабованих середовищах.

4.4.3. Порівняння з комерційними IoT-платформами.

Окрім відкритих та дослідницьких рішень, значну частку ринку займають провідні комерційні екосистеми IoT: Google Home, Amazon Alexa, Apple HomeKit та Samsung SmartThings. Ці платформи забезпечують високий рівень користувацького досвіду, сертифіковані екосистеми пристроїв і централізоване керування через хмарні сервіси виробників. Водночас вони значною мірою залежать від постійного підключення до інтернету та використовують закриті API, що обмежує прозорість, довгострокову підтримуваність і відповідність вимогам щодо конфіденційності.

У порівнянні з такими рішеннями запропонована архітектура на основі Home Assistant має такі ключові переваги:

- Повна локальна автономність: Уся логіка автоматизації, обробка даних і їх зберігання виконуються всередині локальної мережі, забезпечуючи роботу системи навіть під час відсутності інтернету.
- Прозорість відкритого коду: На відміну від закритих платформ, вся конфігурація та

вихідний код системи є доступними для аудиту та модифікації, що гарантує повний контроль користувача.

- Незалежність від обладнання: Платформа інтегрує пристрої різних виробників через Zigbee, Z-Wave, MQTT та REST API, запобігаючи прив'язці до одного постачальника.
- Конфіденційність і відповідність нормативам: Локальний режим роботи відповідає принципам GDPR щодо мінімізації даних і усвідомленої згоди, виключаючи зовнішній профайлінг користувача.
- Економічність і сталість: Система працює на доступному апаратному забезпеченні та повністю відкритому програмному забезпеченні, усуваючи потребу в абонентських платежах і зменшуючи електронні відходи.

Водночас комерційні рішення зберігають певні переваги, зокрема спрощений початковий запуск, наявність офіційної технічної підтримки та широке охоплення сертифікованих пристроїв. Таким чином, запропонована локально орієнтована архітектура позиціонується як комплементарна альтернатива, орієнтована на інституційні та професійні сценарії, де пріоритетами є прозорість, конфіденційність та самостійне управління інфраструктурою.

4.5. Висновки по розділу 4

1. Представлено результати експериментальної перевірки функціональності та ефективності локальної IoT-системи на основі Home Assistant у реальних умовах експлуатації (офісні приміщення площею близько 50 м^2). Показано, що запропонована архітектура забезпечує стабільну безперервну роботу протягом щонайменше семи діб із низьким середнім завантаженням процесора (менше 5 %) та прийнятним використанням оперативної пам'яті при обробці понад 360 000 MQTT-повідомлень. Це підтверджує придатність системи для довготривалої автономної експлуатації в навчальних та офісних середовищах без залежності від зовнішніх хмарних сервісів.
2. Здійснено комплексне порівняння ARM-платформ (класу Raspberry Pi) та x86_64-середовища з гіпервізором Proxmox VE для розгортання локальних IoT-систем на основі реальної інсталяції. Обґрунтовано критичну роль підсистеми зберігання даних та апаратної віртуалізації у забезпеченні продуктивності, надійності та масштабованості. Показано, що x86_64-конфігурація з контейнеризацією сервісів дозволяє стабільно обробляти інтенсивні телеметричні потоки без деградації продуктивності, що робить її придатною для напівпромислових і промислових інсталяцій.
3. Сформульовано та проаналізовано подійно-орієнтовану концепцію автоматизацій у Home Assistant на основі моделі *Trigger–Condition–Action*. Показано, що чітке розділення ролей тригера, умови та дії дозволяє мінімізувати мережевий трафік, зменшити частоту помилкових спрацювань та підвищити передбачуваність поведінки системи. Обґрунтовано, що перехід від циклічного опитування до реакції на події забезпечує стійке функціонування IoT-системи при збільшенні кількості вузлів і складності сценаріїв, що є принципово важливим для побудови масштабованих локальних рішень.
4. Обґрунтовано комплементарне застосування двох підходів до побудови автоматизацій у Home Assistant — візуального редактора та програмної конфігурації у форматі

YAML. Показано, що візуальний редактор є ефективним інструментом для первинного розгортання, прототипування та тестування граничних умов, тоді як YAML-опис забезпечує формалізацію, масштабованість, можливість версіонування та інтеграцію з DevOps-практиками. Комплементарність підходів сприяє поступовій еволюції системи від експериментальної до виробничої.

5. Проведено експериментальний аналіз вибору сенсорів та комунікаційних протоколів (Zigbee, Wi-Fi, MQTT, ESP32-засновані вузли) в контексті вимог до надійності, енергоефективності та автономності локальної IoT-системи. Визначено практичні рекомендації щодо розміщення координаторів і маршрутизаторів Zigbee, подолання бездротової інтерференції, контролю якості зв'язку та управління топологією мережі при зростанні кількості вузлів. Це дозволяє цілеспрямовано проектувати мережеву інфраструктуру для систем з десятками й більшою кількістю пристроїв без втрати стабільності та керованості.
6. На основі методики багатокритеріального оцінювання, розробленої в другому розділі, отримано кількісні результати щодо продуктивності, масштабованості, безпеки, гнучкості та відмовостійкості запропонованої локальної IoT-системи. Показано, що за сукупністю показників система є конкурентоспроможною або перевищує аналоги, включно з комерційними IoT-платформами, за рахунок локалізації обчислень, відсутності залежності від зовнішньої інфраструктури, розширюваності та можливості глибокої кастомізації. Це створює підґрунтя для подальшого використання розробленої архітектури як референсної моделі при впровадженні локальних IoT-рішень у навчальних, офісних та спеціалізованих середовищах.

ВИСНОВКИ

У дисертаційній роботі розв'язано науково-прикладну задачу, пов'язану з розробкою та обґрунтуванням архітектурних, методичних і технічних підходів до побудови локальних автономних IoT-систем моніторингу та керування умовами приміщень. Дослідження виконано з урахуванням сучасних вимог до безпеки, масштабованості, інтеграції гетерогенних пристроїв і використання методів аналітики та машинного навчання в межах приватної інфраструктури. Основні результати дисертаційної роботи полягають у наступному.

1. Виконано комплексний аналіз сучасних архітектурних моделей, протоколів і програмних платформ IoT, у результаті чого виявлено обмеження хмарноорієнтованих рішень у контексті автономності, безпеки та відтворюваності експериментів. Це дозволило науково обґрунтувати доцільність переходу до локальних автономних IoT-архітектур як основи для побудови надійних і контрольованих систем моніторингу та керування.
2. Запропоновано багаторівневу архітектуру локальної IoT-системи, що охоплює фізичний, комунікаційний, інформаційний, керуючий, безпековий, сервісний і презентаційний рівні. Така архітектура забезпечує чітке розділення функцій, підвищує модульність і масштабованість системи та створює передумови для її адаптації до різних сценаріїв експлуатації без втрати цілісності.
3. Розроблено функціональну модель локальної IoT-системи на базі платформи Home Assistant, яка забезпечує інтеграцію гетерогенних пристроїв, локальне виконання сценаріїв автоматизації та централізоване керування без залежності від зовнішніх хмарних сервісів. Це дає змогу реалізувати автономну систему з мінімальними затримками та повним контролем над даними.
4. Удосконалено підхід до забезпечення кібербезпеки локальних IoT-систем шляхом наскрізної інтеграції механізмів мережевої сегментації, TLS/SSL-шифрування, контролю доступу, журналювання подій і багатофакторної автентифікації. Запропонований комплексний рівневий захист дозволяє суттєво знизити ризики витоку даних, несанкціонованого доступу та порушення працездатності системи.
5. Розроблено методику багатокритеріальної оцінки ефективності локальної IoT-системи, яка передбачає інтеграцію показників продуктивності, надійності, інформаційної безпеки та енергоспоживання в узагальнений критерій. Це забезпечує можливість об'єктивного порівняння різних конфігурацій системи та прийняття обґрунтованих рішень щодо її оптимізації та подальшого розвитку.
6. Удосконалено механізми керування пристроями та сценаріями автоматизації за рахунок використання шаблонізованих конфігурацій, тригерно-орієнтованої логіки та розділення рівнів прийняття рішень. Такий підхід підвищує адаптивність системи, зменшує ймовірність помилок конфігурації та спрощує супровід і масштабування в процесі експлуатації.
7. Сформовано сервісний рівень у вигляді уніфікованого API-шару з підтримкою REST та

WebSocket протоколів, що забезпечує двосторонню асинхронну взаємодію в реальному часі. Це створює можливість інтеграції локальної IoT-системи з зовнішніми аналітичними модулями та програмними сервісами без втрати безпеки й продуктивності.

8. Виконано експериментальну оцінку ефективності розробленої локальної IoT-системи в умовах реального функціонування, у результаті чого підтверджено її працездатність, низькі затримки обробки подій, стабільність передачі даних та здатність до автономного функціонування без критичної залежності від зовнішньої хмарної інфраструктури. Отримані результати підтвердили ефективність запропонованих архітектурних і програмних рішень щодо забезпечення надійності, масштабованості та безпеки системи.
9. Обґрунтовано доцільність інтеграції методів аналітики даних і машинного навчання в локальні IoT-системи з використанням edge-орієнтованих підходів. Реалізація локальної обробки даних дозволяє зменшити затримки, підвищити стійкість системи до мережевих збоїв та забезпечити збереження конфіденційності інформації в межах приватної інфраструктури.

Отримані в дисертаційній роботі результати мають наукову новизну та практичну цінність і можуть бути використані під час проєктування та впровадження локальних автономних IoT-систем у задачах моніторингу, автоматизації та інтелектуального керування умовами середовища, а також у подальших наукових дослідженнях у галузі кіберфізичних систем і Internet of Things.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Khomenko Y. and Babichev S. Contemporary Methods, Models, and Software for Implementation and Optimization of IoT (Internet of Things) Systems: A Review // Lecture Notes in Data Engineering, Computational Intelligence, and Decision-Making, Volume 2. ISDMCI 2024 / ed. by Babichev S. and Lytvynenko V. Springer, Cham, 2025. Vol. 244 of *Lecture Notes on Data Engineering and Communications Technologies*. P. 134–158.

[2] Хоменко Є. Сучасні методи, моделі та програмні засоби реалізації та оптимізації систем IoT (INTERNET OF THINGS) // *Information Technologies in Education (ITE)*. 2024. № 55. С. 72–84.

[3] Khomenko Ye. i Babichev S. Advantages and disadvantages of modern Internet of Things systems // Матеріали XX Міжнародної наукової інтернет-конференції «Intellectual Systems for Decision Making and Problems of Computational Intelligence». м. Усті на Лабі, Чехія. 2024. С. 41–44.

[4] Khomenko Y. Comparison of commercial and local IoT systems for environmental monitoring in educational institutions // Abstracts of XIX International Scientific and Practical Conference. Krakow, Poland. 2025. P. 228–234.

[5] Taha M.F., ElMasry G., Gouda M., Zhou L., Liang N., Abdalla A., Rousseau D., and Qiu Z. Systems and Internet of Things (IoT) for Aquaponics Automation: A Comprehensive Overview // *Chemosensors*. 2022. Vol. 10. Art. no. 303.

[6] Himmat M., Algazoli G., Hammam N., and Abdalla A. Review on the Current State of the Internet of Things and its Extension and its Challenges // *European Journal of Information Technologies and Computer Science*. 2022. Vol. 2. P. 1–5.

[7] Laroui Mohammed, Nour Boubakr, Moungla Hassine, Cherif Moussa A., and Afifi Hossam. Edge and fog computing for IoT: A survey on current research activities & future directions // *Computer Communications*. 2021. Vol. 180. P. 210–231.

[8] Philip Sumesh J., Luu Truong (Jack), and Carte Traci. There's No place like home: Understanding users' intentions toward securing internet-of-things (IoT) smart home networks // *Computers in Human Behavior*. 2023. Vol. 139. P. 107551.

[9] Chegini H. and Mahanti A. A Framework of Automation on Context-Aware Internet of Things (IoT) Systems // Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion. New York, NY, USA : Association for Computing Machinery. 2019. P. 157–162.

[10] Soni T., Gupta D., Uppal M., Dutta M., and Sharma A. Automation in Fog Cloud assisted Internet of Things Ecosystem: Challenges, Components and Protocols // 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA). 2023. P. 313–317.

[11] Firouzi Farshad, Farahani Bahar, and Marinšek Alexander. The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT) // *Information Systems*. 2022. Vol. 107. P. 101840.

[12] Hammi Badis, Zeadally Sherali, Khatoun Rida, and Nebhen Jamel. Survey on smart homes: Vulnerabilities, risks, and countermeasures // *Computers & Security*. 2022. Vol. 117. P. 102677.

[13] Yar Hikmat, Imran Ali Shariq, Khan Zulfiqar Ahmad, Sajjad Muhammad, and Kastrati Zenun. Towards Smart Home Automation Using IoT-Enabled Edge-Computing Paradigm // *Sensors*. 2021. Vol. 21, no. 14. Art. no. 4932.

[14] Hamdan S., Ayyash M., and Almajali S. Edge-Computing Architectures for Internet of Things Applications: A Survey // *Sensors*. 2020. Vol. 20, no. 22. Art. no. 6441.

[15] Addow M. A., Jimale A. D., Yasin Nageye A., Abdullahi M. O., and Hilowle S. M. A Low-Cost IoT-Based Smart Home Automation System for Urban Sustainability in Mogadishu, Somalia // *Discover Internet of Things*. 2025. Vol. 5. Art. no. 99.

[16] Taiwo O. and Ezugwu A. E. Internet of Things-Based Intelligent Smart Home Control System // *Security and Communication Networks*. 2021. Vol. 1. Art. no. 9928254.

[17] Antolini A., Zavalloni F., Lico A., Quqa S., Greco L., Mangia M., Pareschi F., Pasotti M., and Franchi Scarselli E. The Role of Phase-Change Memory in Edge Computing and Analog In-Memory Computing: An Overview of Recent Research Contributions and Future Challenges // *Sensors*. 2025. Vol. 25, no. 12. Art. no. 3618.

[18] Xie Y. and Fang Q. An Energy-Aware Generative AI Edge Inference Framework for Low-Power IoT Devices // *Electronics*. 2025. Vol. 14, no. 20. Art. no. 4086.

[19] Aavild A.P., Rosenkrantz de Lasson A., Moesgaard Andersen Ch., and et al. Distributed Home Automation with Home Assistant // Proceedings of the 14th International Conference on the Internet of Things. New York, NY, USA : Association for Computing Machinery. 2025. P. 206–212.

[20] Home Assistant Community. Home Assistant. Available at: https://www.home-assistant.io/ (accessed May 2025).

[21] Cujilema Paguay J.A., Hidalgo Brito G.A., Hernandez Rojas D.L., and Cartuche Calva J.J. Secure home automation system based on ESP-NOW mesh network, MQTT and Home Assistant platform // *IEEE Latin America Transactions*. 2023. Vol. 21, no. 7. P. 829 – 838.

[22] Sita I.V. and David B. Development and Implementation of a Comprehensive Smart Home Automation System Using Home Assistant and IoT Devices // 2024 4th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). 2024. P. 1–7.

[23] Oulefki A., Amira A., Kurugollu F., and Alshoweky M. Twining Buildings: A Methodological Framework for Design and Implementation using Home Assistant Technology // 2023 International Conference on Electrical, Communication and Computer Engineering (ICECCE). 2023. P. 1–6.

[24] Akhmetzhanov B. K., Gazizuly O. A., Nurlan Z., and Zhakiyev N. Integration of a Video Surveillance System Into a Smart Home Using the Home Assistant Platform // 2022 Interna-

tional Conference on Smart Information Systems and Technologies (SIST). 2022. P. 1–5.

[25] Corbett C.F, Combs E.M., Wright P.J., Owens O.L., I. Stringfellow, Nguyen T., and Van Son C.R. Virtual Home Assistant Use and Perceptions of Usefulness by Older Adults and Support Person Dyads // *International Journal of Environmental Research and Public Health*. 2021. Vol. 18, no. 3. Art. no. 1113.

[26] Tsakalidis S., Tsoulos G., Kontaxis D., and Athanasiadou G. Design and Implementation of a Versatile OpenHAB IoT Testbed with a Variety of Wireless Interfaces and Sensors // *Telecom*. 2023. Vol. 4. P. 597–610.

[27] openHAB Foundation. openHAB – Empowering the Smart Home. Available at: https://www.openhab.org/ (accessed May 2025).

[28] Pellegrino P., Bonino D., and Corno F. Domotic house gateway // *Proceedings of the 2006 ACM symposium on Applied computing*. 2006. Access mode: https://api.semanticscholar.org/CorpusID:18661387.

[29] Domoticz Contributors. Domoticz – Home Automation System. Available at: https://www.domoticz.com/ (accessed May 2025).

[30] Valencia-Arias Alejandro, Cardona-Acevedo Sebastián, Gómez-Molina Sergio, Gonzalez-Ruiz Juan David, and Valencia Jackeline. Smart home adoption factors: A systematic literature review and research agenda // *PLOS ONE*. 2023. Vol. 18, no. 10. P. e0292558.

[31] Risteska-Stojkoska B. and Trivodaliev K. A review of Internet of Things for smart home: Challenges and solutions // *Journal of Cleaner Production*. 2017. Vol. 140. P. 1454–1464.

[32] Andriulo Francesco Cosimo, Fiore Marco, Mongiello Marina, Traversa Emanuele, and Zizzo Vera. Edge Computing and Cloud Computing for Internet of Things: A Review // *Informatics*. 2024. Vol. 11, no. 4. P. 71.

[33] Mota Afonso, Serôdio Carlos, and Valente António. Matter Protocol Integration Using Espressif's Solutions to Achieve Smart Home Interoperability // *Electronics*. 2024. Vol. 13, no. 11. P. 2217.

[34] Ansari D.B., Rehman A., and Ali R. Internet of Things (IoT) Protocols: A Brief Exploration of MQTT and CoAP // *International Journal of Computer Applications*. 2018. Vol. 179. P. 9–14.

[35] Chui K., Gupta B.B., Liu J., and et al. A Survey of Internet of Things and Cyber-Physical Systems: Standards, Algorithms, Applications, Security, Challenges, and Future Directions // *Information*. 2023. Vol. 14. P. art. no. 388.

[36] Esposito Marco, Belli Alberto, Palma Lorenzo, and Pierleoni Paola. Design and Implementation of a Framework for Smart Home Automation Based on Cellular IoT, MQTT, and Serverless Functions // *Sensors*. 2023. Vol. 23, no. 9. P. 4459.

[37] Gubbi Jayavardhana, Buyya Rajkumar, Marusic Slaven, and Palaniswami Marimuthu. Internet of Things (IoT): A vision, architectural elements, and future directions // *Future Generation Computer Systems*. 2013. Vol. 29, no. 7. P. 1645–1660.

[38] Zanella Andrea, Bui Nicola, Castellani Angelo, Vangelista Lorenzo, and Zorzi Michele. Internet of Things for Smart Cities // *IEEE Internet of Things Journal*. 2014. Vol. 1, no. 1. P. 22–32.

[39] Borgia Eleonora. The Internet of Things vision: Key features, applications and open issues // *Computer Communications*. 2014. Vol. 54. P. 1–31.

[40] Trigka Maria and Dritsas Elias. Edge and Cloud Computing in Smart Cities // *Future Internet*. 2025. Vol. 17, no. 3. P. 118.

[41] Grobelna Iwona. Internet of Things and Cyber–Physical Systems // *Future Internet*. 2022. Vol. 14, no. 11. P. 337.

[42] FakhrHosseini Shabnam, Lee Chaiwoo, Lee Sheng-Hung, and Coughlin Joseph. A Taxonomy of Home Automation: Expert Perspectives on the Future of Smarter Homes // *Information Systems Frontiers*. 2025. Vol. 27. P. 449–466.

[43] Survey on Smart Home Users' Security and Privacy Perceptions and Actions: A Device Category Perspective : NIST Special Publication : 1343 / National Institute of Standards and Technology (NIST) ; executor: Haney Julie, Acar Yasemin, Li Anna, and Haney Faith : 2025.

[44] Donta Praveen Kumar, Srirama Satish Narayana, Amgoth Tarachand, and Annavarapu Chandra Sekhara Rao. Survey on recent advances in IoT application layer protocols and machine learning scope for research directions // *Digital Communications and Networks*. 2022. Vol. 8, no. 5. P. 727–744.

[45] Nedergaard Kristofer, Singh Bhupjit, and Andersen Birger. Evaluating CoAP, OSCORE, DTLS and HTTPS for Secure Device Communication // Internet of Everything. The First EAI International Conference, IoECon 2022, Guimaräes, Portugal, September 16–17, 2022, Proceedings. 2023. P. 125–137.

[46] Fakhruldeen H. F., Saadh M., Khan S., and et al. Enhancing smart home device identification in WiFi environments for futuristic smart networks-based IoT // *International Journal of Data Science and Analytics*. 2024. P. 1–14.

[47] Romputtal A. and Phongcharoenpanich C. T-Slot Antennas-Embedded ZigBee Wireless Sensor Network System for IoT-Enabled Monitoring and Control Systems // *IEEE Internet of Things Journal*. 2023. Vol. 10, no. 23. P. 20834–20845.

[48] Gunnarsson Martin, Brorsson Joakim, Palombini Francesca, Seitz Ludwig, and Tiloca Marco. Evaluating the performance of the OSCORE security protocol in constrained IoT environments // *Internet of Things*. 2021. Vol. 13. P. 100333.

[49] Puthiyidam J. IoT Smart Home: Protocols and Architectures // *International Journal for Research in Applied Science and Engineering Technology*. 2017. Vol. 5. P. 2031–2038.

[50] Piñeres Espitia Gabriel, Butt Shariq Aziz, Estévez-Ortiz Francisco, Cama-Pinto Alejandro, and Maleh Yassine. Performance analysis of 6LoWPAN protocol for a flood monitoring system // *EURASIP Journal on Wireless Communications and Networking*. 2022. Vol. 2022, no. 16.

[51] Prajapati Anil K., Pilli Emmanuel S., Battula Ramesh B., Varadharajan Vijay, Verma Abhishek, and Joshi R. C. A comprehensive survey on RPL routing-based attacks, defences and future directions in Internet of Things // *Computers and Electrical Engineering*. 2025. Vol. 123. P. 110071.

[52] Guide to Bluetooth Security : NIST Special Publication : 800-121 Revision 2 / National Institute of Standards and Technology (NIST) ; executor: Padgette John, Bahr John, Batra Mayank, Holtmann Marcel, Smithbey Rhonda, Chen Lily, and Scarfone Karen : 2022.

[53] Song Eugene, da Rocha Helbert, Espirito-Santo Antonio, and Brama Riccardo. IEEE 1451.0-based Web of Thing (WoT) Ontology // Proceedings of The 2024 Annual Conference of the IEEE Industrial Electronics Society (IECON). 2024.

[54] Al-Shareeda M., Ali M., Manickam S, and Karuppayah S. Bluetooth low energy for internet of things: review, challenges, and open issues Bluetooth low energy Bluetooth low energy for internet of things Internet of things // *Indonesian Journal of Electrical Engineering and Computer Science*. 2023. Vol. 31. P. 1182–1189.

[55] Khattak S., Nasralla M., Farman H., and Choudhury N. Performance Evaluation of an IEEE 802.15.4-Based Thread Network for Efficient Internet of Things Communications in Smart Cities // *Applied Sciences*. 2023. Vol. 13. P. art. no. 7745.

[56] Loukil Slim, Fourati Lamia Chaari, Nayyar Anand, and Chee K.-W.-A. Analysis of LoRaWAN 1.0 and 1.1 Protocols Security Mechanisms // *Sensors*. 2022. Vol. 22, no. 10. P. 3717.

[57] Kumar Shashwat, Hahn Francis, Ou Xinming, and Singhal Anoop. Security Analysis of Trust on the Controller in the Matter Protocol // 2023 IEEE Conference on Communications and Network Security (CNS). 2023. P. 1–6.

[58] Cäsar Matthias, Pawelke Tobias, Steffan Jan, and Terhorst Gabriel. A survey on Bluetooth Low Energy security and privacy // *Computer Networks*. 2022. Vol. 205. P. 108712.

[59] Ghobakhlou Akbar, Al-Hamid Duaa Zuhair, Zandi Sara, and Cato James. A Comprehensive Analysis of Security Challenges in ZigBee 3.0 Networks // *Sensors*. 2025. Vol. 25, no. 15. P. 4606.

[60] Akshatha P. S., Dilip Kumar S. M., and Venugopal K. R. MQTT Implementations, Open Issues, and Challenges: A Detailed Comparison and Survey // *International Journal of Sensors, Wireless Communications and Control*. 2022. Vol. 12, no. 8. P. 553–576.

[61] Hue Axelle, Sharma Gaurav, and Dricot Jean-Michel. Privacy-Enhanced MQTT Protocol for Massive IoT // *Electronics*. 2022. Vol. 11, no. 1. P. 70.

[62] Al Hanif Abdulelah and Ilyas Mohammad. Effective Feature Engineering Framework for Securing MQTT Protocol in IoT Environments // *Sensors*. 2024. Vol. 24, no. 6. P. 1782.

[63] Singh S. Intercompatibility of IoT Devices Using Matter: Next-Generation IoT Connectivity Protocol // Advances in IoT and Security with Computational Intelligence. 2023. P. 49–58. ISBN: 978-981-99-5087-4.

[64] Gupta N., Juneja P., Sharma S., and Garg U. An intelligent technique for network resource management and analysis of 5G-IoT smart healthcare application // *Journal of Autonomous Intelligence*. 2023. Vol. 7, no. 1. P. art. no. 694.

[65] Farhad Arshad and Pyun Jae-Young. Resource Management for Massive Internet of Things in IEEE 802.11ah WLAN: Potentials, Current Solutions, and Open Challenges // *Sensors*. 2022. Vol. 22, no. 23. P. 9509.

[66] Taramit Hamid, Orozco-Barbosa Luis, and Haqiq Abdelkrim. A renewal theory based performance and configuration framework of the IEEE 802.11ah RAW mechanism // *Digital Communications and Networks*. 2023. Vol. 9, no. 1. P. 236–251.

[67] Ugwuanyi Stephen, Paul Greig, and Irvine James. Survey of IoT for Developing Countries: Performance Analysis of LoRaWAN and Cellular NB-IoT Networks // *Electronics*. 2021.

Vol. 10, no. 18. P. 2224.

[68] Ogbodo Emmanuel U., Abu-Mahfouz Adnan M., and Kurien Anish. A Survey on 5G and LPWAN-IoT for Improved Smart Cities and Remote Area Applications: From the Aspect of Architecture and Security // *Sensors*. 2022. Vol. 22, no. 16. P. 6313.

[69] Muhammad Siraj, Zhao Jiamiao, and Refai Hazem H. An Empirical Analysis of IEEE 802.11ax // 2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA). 2021. P. 1–6.

[70] Xu Yanli and Huang Jinhui. A Survey on Time-Sensitive Networking Standards and Applications for Intelligent Driving // *Processes*. 2023. Vol. 11, no. 7. P. 2211.

[71] Ji Yahui, Yu Jiguo, Yao Yan, Yu Kan, Chen Honglong, and Zheng Shanchao. Securing wireless communications from the perspective of physical layer: A survey // *Internet of Things*. 2022. Vol. 19. P. 100524.

[72] Abdi Abdinasir Hirsi, Audah Lukman, Salh Adeb, and et al. Artificial intelligence performance evaluation for URLLC of industrial IoT applications: A review, open challenges and future directions // *Physical Communication*. 2025. Vol. 72. P. 102712.

[73] Shayea Ibraheem, El-Saleh Ayman A., Ergen Mustafa, Saoud Bilal, Hartani Riad, Turan Derya, and Kabbani Adnan. Integration of 5G, 6G and IoT with Low Earth Orbit (LEO) networks: Opportunity, challenges and future trends // *Results in Engineering*. 2024. Vol. 23. P. 102409.

[74] Alves Hirley, Jo Gweon Do, Shin JaeSheung, and et al. Beyond 5G URLLC evolution: New service modes and practical considerations // *ITU Journal on Future and Evolving Technologies*. 2022. Vol. 3, no. 3. P. 545–554.

[75] Crawford Colin. Protocol power: Matter, IoT interoperability, and a critique of industry self-regulation // *Internet Policy Review*. 2024. Vol. 13, no. 2.

[76] Santos Andre, Gonçalves Rui, Pérez Javier, and et al. FIWARE IoT agent for Matter: Toward the integration of smart home devices into the FIWARE smart city platform // *Computer Communications*. 2026. Vol. 245. P. 108369.

[77] Du Hua, Han Qi, Yang Dujuan, de Vries Bauke, and van Houten Thomas. Data privacy and smart home energy appliances: A stated choice experiment // *Heliyon*. 2023. Vol. 9, no. 11. P. e21448.

[78] Zegeye Wondimu, Jemal Ahamed, and Kornegay Kevin. Connected Smart Home over Matter Protocol // 2023 IEEE International Conference on Consumer Electronics (ICCE). 2023. P. 1–7.

[79] Li Xin, Zhang Yifan, Wang Haoran, and et al. A survey on privacy and security issues in IoT-based environments: Technologies, protection measures and future directions // *Computers & Security*. 2025. Vol. 148. P. 104097.

[80] Buil-Gil David, Kemp Steven, Kuenzel Stefanie, Coventry Lynne, Zakhary Sameh, Tilley Daniel, and Nicholson James. The digital harms of smart home devices: A systematic literature review // *Internet of Things*. 2023.

[81] Shetty A., Ketan K., and Muniandi S. Embedded Web Server Application for Industrial Automation // *Indian Journal of Science and Technology*. 2015. Vol. 8. P. art. no. 235.

[82] Manivannan M. and Kumaresan N. Design of on-line Interactive Data Acquisition and Control System for embedded real time applications. 2011. P. 551–556.

[83] Nuriev M. G., Kremleva E. S., Pikuleva N. I., and Khafizova A. S. Development of a smart home control system based on Home Assistant and Zigbee // *International Research Journal*. 2025. no. 7 (157). P. 11.

[84] Moure-Garrido Marta, Garcia-Rubio Carlos, and Campo Celeste. Reducing DNS Traffic to Enhance Home IoT Device Privacy // *Sensors*. 2024. Vol. 24, no. 9. P. 2690.

[85] Reis Manuel J. C. S. and Serôdio Carlos. Edge AI for Real-Time Anomaly Detection in Smart Homes // *Future Internet*. 2025. Vol. 17, no. 4. P. 179.

[86] Estrada A. and et al. An open source IoT edge-computing system for monitoring energy consumption in buildings // *Results in Engineering*. 2024. Vol. 21. P. 101875.

[87] How intelligence is reshaping today: IoE edge networks? // *Internet of Things*. 2025. Vol. 33. P. 101717.

[88] A dynamic method to protect user privacy against traffic-based attacks on smart home // *Ad Hoc Networks*. 2023. Vol. 149. P. 103226.

[89] Jointly Achieving Smart Homes Security and Privacy through Bidirectional Trust // *EURASIP Journal on Information Security*. 2025. Vol. 2025, no. 17.

[90] Home Assistant. 2024. Access mode: https://www.home-assistant.io/.

[91] Thakor J. Home Automation over the Home Assistant using Raspberry Pi // *International Journal for Research in Applied Science and Engineering Technology*. 2020. Vol. 8. P. 359–365.

[92] Jain R. Getting Started with Home Assistant // Advanced Home Automation Using Raspberry Pi: Building Custom Hardware, Voice Assistants, and Wireless Nodes. 2021. P. 163–210. ISBN: 978-1-4842-7274-9.

[93] Marius C. S., Laurențiu M., and Cristian R. C. Optimal Monitoring of Server Rooms with Home Assistant Platform // *MATEC Web of Conferences*. 2022. Access mode: https://api.semanticscholar.org/CorpusID:254950130.

[94] Bylieva D., Bekirogullari Z., Lobatyuk V.a, and Anosova N. Home Assistant of The Future: What is It Like? // Proceedings of the International Scientific Conference - Digital Transformation on Manufacturing, Infrastructure and Service. Association for Computing Machinery. 2021. P. art. no. 60.

[95] Iftimie F. and Vinte C. Upon a Home Assistant Solution Based on Raspberry Pi Platform // *Informatica Economica*. 2017. Vol. 21. P. 5–16.

[96] Home Assistant Install. 2024. Access mode: https://www.home-assistant.io/installation/.

[97] Thomas-Krenn-Award 2019 – Die Gewinner. 2024. Access mode: https://www.thomas-krenn.com/de/tkmag/tk-insights/thomas-krenn-award-2019-gewinner/.

[98] Das sind die Nominierten für die Dinacon Awards 2018. 2024. Access mode: https://www.netzwoche.ch/news/2018-09-04/das-sind-die-nominierten-fuer-die-dinacon-awards-2018.

[99] Aboulhir M., Khoulji S., Jourani R., and Kerkeb M. Integration of the ASR Toolkit Kaldi Into a Domoticz Home Automation System // *Transactions on Engineering and Computing Sciences*. 2017. Vol. 5, no. 4.

[100] de Oliveira G. A. A., de Bettio R. W., and Freire A. P. Accessibility of the smart home for users with visual disabilities: an evaluation of open source mobile applications for home automation // Proceedings of the 15th Brazilian Symposium on Human Factors in Computing Systems. Association for Computing Machinery. 2016. P. art. no. 29.

[101] Kryczka C. Own system of intelligent building in comparison with an open - source solution // *Journal of Computer Sciences Institute*. 2019. Vol. 12. P. 232–239.

[102] Redzwan F. N. M., Suhaimi A. H. F., and Husaini A. K. The development of IoT Smart House Automation System controlled wirelessly via OpenHab central server // *IOP Conference Series: Materials Science and Engineering*. 2021. Vol. 1176, no. 1. P. art. no. 012021.

[103] Borissova D., Danev V., Rashevski M., Garvanov I., and et al. Using IoT for Automated Heating of a Smart Home by Means of OpenHAB Software Platform // *IFAC-PapersOnLine*. 2022. Vol. 55, no. 11. P. 90–95. IFAC Workshop on Control for Smart Cities CSC 2022.

[104] Tsakalidis S., Tsoulos G., Kontaxis D., and Athanasiadou G. Design and Implementation of a Versatile OpenHAB IoT Testbed with a Variety of Wireless Interfaces and Sensors // *Telecom*. 2023. Vol. 4, no. 3. P. 597–610.

[105] Davitadze Z., Kakhiani G., and Pasieshvili D. Remotely Controlled Experiments on the Basis of Raspberry Pi and openHAB // 2021 IEEE East-West Design and Test Symposium (EWDTS). 2021. P. 1–4.

[106] Saha S. and et al. A Survey on Privacy Preservation Techniques in IoT Systems // *Sensors*. 2025. Vol. 25, no. 22. P. 6967.

[107] Khan M. and et al. A Review of IoT Firmware Vulnerabilities and Auditing Techniques // *Sensors*. 2024. Vol. 24, no. 2. P. 708.

[108] Pham Le Phuc-Hung, Do Quy Ngoc, Dinh Toan Q., Nguyen Luong Vuong, and Pham Ho-Trong-Nguyen. A comparative security analysis of MQTT brokers against DoS attacks // *Journal on Information Security*. 2026.

[109] Survey on Smart Home Users' Security and Privacy Perceptions and Actions: A Device Category Perspective : NIST Special Publication : 1343 / National Institute of Standards and Technology (NIST) ; executor: Haney Julie, Acar Yasemin, Li Anna, and Haney Faith : 2025.

[110] Maaloul N. and et al. Software vulnerability management in IoT systems: a systematic mapping study // *Cybersecurity*. 2026. Vol. 9, no. 96.

[111] Gane A. and et al. Securing MQTT: unveiling vulnerabilities and innovating cyber range solutions // *Procedia Computer Science*. 2024. Vol. 241. P. 69–76.

[112] Li Y. and et al. A security-enhanced scheme for MQTT protocol based on domestic cryptographic algorithm // *Computer Communications*. 2024. Vol. 221. P. 1–9.

[113] Soni A. and et al. Blockchain for Secure IoT: A Review of Identity Management, Access Control, and Trust Mechanisms // *IoT*. 2025. Vol. 6, no. 4. P. 65.

[114] Hosseinzadeh Ali and Chen F. Frank. A Two-Stage Hybrid Federated Learning Framework for Privacy-Preserving IoT Anomaly Detection and Classification // *IoT*. 2025. Vol. 6, no. 3. P. 48.

[115] Birkmose Rune, Reece Nathan Morkeberg, Norvin Esben Hofstedt, Bjerva Johannes, and Zhang Mike. On-Device LLMs for Home Assistant: Dual Role in Intent Detection and Response

Generation // *ArXiv*. 2025. Vol. abs/2502.12923. Access mode: https://api.semanticscholar.org/CorpusID:276421416.

[116] Heydari Soroush and Mahmoud Qusay H. Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions // *Sensors*. 2025. Vol. 25, no. 10. P. 3191.

[117] Oliveira Franklin, Costa Daniel G., Assis Flávio, and Silva Ivanovitch. Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning // *Internet of Things*. 2024. Vol. 26. P. 101153.

[118] Gad Mark M., Gad Walaa, Abdelkader Tamer, and Naik Kshirasagar. Personalized Smart Home Automation Using Machine Learning: Predicting User Activities // *Sensors*. 2025. Vol. 25(19). P. 6082. Access mode: https://doi.org/10.3390/s25196082.

[119] Li Yunlong, Zhang Rongguang, Zhao Pengcheng, and Wei Yunkai. Feature-Attended Federated LSTM for Anomaly Detection in the Financial Internet of Things // *Applied Sciences*. 2024. Vol. 14, no. 13. P. 5555.

[120] Villegas-Ch William, Govea Jaime, and Jaramillo-Alcazar Angel. IoT Anomaly Detection to Strengthen Cybersecurity in the Critical Infrastructure of Smart Cities // *Applied Sciences*. 2023. Vol. 13(19). P. 10977. Access mode: https://doi.org/10.3390/app131910977.

[121] Hendaoui Fatma, Meddeb Rahma, Trabelsi Lamia, Ferchichi Ahlem, and Ahmed Rawia. FLADEN: Federated Learning for Anomaly DEtection in IoT Networks // *Computers & Security*. 2025. Vol. 155. P. 104446.

[122] Karuna G., Ediga P., Akshatha S., Anupama P., Sanjana T., Mittal A., Rajvanshi S., and Habelalmateen M.I. Smart Energy Management: Real-Time Prediction and Optimization for IoT-Enabled Smart Homes // *Cogent Engineering*. 2024. Vol. 11(1). P. 2390674. Access mode: https://doi.org/10.1080/23311916.2024.2390674.

[123] Gayathri S. and Surendran D. Unified ensemble federated learning with cloud computing for online anomaly detection in energy-efficient wireless sensor networks // *Journal of Cloud Computing*. 2024. Vol. 13, no. 49.

[124] Yildiz Onem and Sucuoglu Hilmi Saygin. Development of Real-Time IoT-Based Air Quality Forecasting System Using Machine Learning Approach // *Sustainability*. 2025. Vol. 17(19). P. 8531. Access mode: https://doi.org/10.3390/su17198531.

[125] Andriulo Francesco Cosimo, Fiore Marco, Mongiello Marina, Traversa Emanuele, and Zizzo Vera. Edge Computing and Cloud Computing for Internet of Things: A Review // *Informatics*. 2024. Vol. 11(4). P. 71. Access mode: https://doi.org/10.3390/informatics11040071.

[126] Aavild A.P., Rosenkrantz de Lasson A., Moesgaard Andersen Ch., and et al. Distributed Home Automation with Home Assistant // Proceedings of the 14th International Conference on the Internet of Things. New York, NY, USA : Association for Computing Machinery. 2025. P. 206–212.

[127] Khomenko Y. and Babichev S. Modular IoT Architecture for Monitoring and Control of Office Environments Based on Home Assistant // *IoT*. 2025. Vol. 6, no. 4. P. 69.

[128] Babichev S., Yarema O., Khomenko Y., Senchyshen D., and Durnyak B. Sensor-Oriented Framework for Underwater Acoustic Signal Classification Using EMD–Wavelet Filtering and

Bayesian-Optimized Random Forest // *Sensors*. 2025. Vol. 25, no. 17. P. 5336.

[129] Хоменко Є. В. і Бабічев С. А. Автономна IoT-система моніторингу мікроклімату аудиторій на основі відкритої DIY-архітектури // *Прикладні питання математичного моделювання*. 2025. Т. 8, № 1. С. 245–254.

[130] Senchyshen D. and Khomenko Y. Integration of IoT sensors into the KSU24 digital learning environment: infrastructure and pedagogical implications // Abstracts of XX International Scientific and Practical Conference. Plovdiv, Bulgaria. 2025. P. 222–226.

[131] Khomenko Y. Modern voice assistants of intellectual systems and their use in Internet of Things (IoT) systems // Abstracts of XXII International Scientific and Practical Conference. Krakow, Poland. 2025. P. 208–214.

[132] Хоменко Є. і Лемещук О. Технічні аспекти та норми для систем керування «розумний будинок» в умовах війни // Матеріали IV International Scientific and Theoretical Conference «Features of the development of modern science in the pandemic's era». Berlin, Germany. 2023. С. 73–75.

[133] Atzori Luigi, Iera Antonio, and Morabito Giacomo. The Internet of Things: A survey // *Computer Networks*. 2010. Vol. 54, no. 15. P. 2787–2805.

[134] Gubbi Jayavardhana, Buyya Rajkumar, Marusic Slaven, and Palaniswami Marimuthu. Internet of Things (IoT): A vision, architectural elements, and future directions // *Future Generation Computer Systems*. 2013. Vol. 29, no. 7. P. 1645–1660.

[135] Ansari Danish Bilal, Rehman Atteeq-Ur, and Ali Rizwan. Internet of Things (IoT) Protocols: A Brief Exploration of MQTT and CoAP // *International Journal of Computer Applications*. 2018. 03. Vol. 179. P. 9–14.

[136] Puthiyidam Jiby. IoT Smart Home: Protocols and Architectures // *International Journal for Research in Applied Science and Engineering Technology*. 2017. 11. Vol. V. P. 2031–2038.

[137] Chui Kwok, Gupta Brij B, Liu Jiaqi, Arya Varsha, Nedjah Nadia, Almomani Dr.Ammar, and Chaurasia Priyanka. A Survey of Internet of Things and Cyber-Physical Systems: Standards, Algorithms, Applications, Security, Challenges, and Future Directions // *Information*. 2023. 07. Vol. 14. P. 388.

[138] Fakhruldeen Hassan Falah, Saadh Mohamed, Khan Samiullah, Salim Nur, Jhamat Naveed, and Mustafa Ghulam. Enhancing smart home device identification in WiFi environments for futuristic smart networks-based IoT // *International Journal of Data Science and Analytics*. 2024. 01. P. 1–14.

[139] Romputtal Adisak and Phongcharoenpanich C. T-Slot Antennas-Embedded ZigBee Wireless Sensor Network System for IoT-Enabled Monitoring and Control Systems // *IEEE Internet of Things Journal*. 2023. 12. Vol. PP. P. 1–1.

[140] Let's Encrypt. Let's Encrypt – Free SSL/TLS Certificates. Available at: https://letsencrypt.org (accessed May 2025).

[141] Shi Weisong and Dustdar Schahram. The Promise of Edge Computing // *Computer*. 2016. Vol. 49, no. 5. P. 78–81.

[142] Home Assistant Community. Home Assistant. 2024. Access mode: https://www.home-assistant.io/. Accessed: May 2025.

[143] Proxmox Server Solutions GmbH. Proxmox Virtual Environment. 2024. Access mode: https://www.proxmox.com/en/proxmox-ve. Accessed: May 2025.

[144] Hat Red. KVM - Kernel-based Virtual Machine. 2024. Access mode: https://www.linux-kvm.org/. Accessed: May 2025.

[145] LinuxContainers.org. Linux Containers (LXC). 2024. Access mode: https://linuxcontainers.org/. Accessed: May 2025.

[146] Open OASIS. MQTT Protocol. 2024. Access mode: https://mqtt.org/. Accessed: May 2025.

[147] Alliance Connectivity Standards. Zigbee Alliance. 2024. Access mode: https://zigbeealliance.org/. Accessed: May 2025.

[148] Alliance Z-Wave. Z-Wave Protocol. 2024. Access mode: https://z-wavealliance.org/. Accessed: May 2025.

[149] Thread Group. Thread Group - Secure and Reliable Networking Protocol. 2024. Access mode: https://www.threadgroup.org/. Accessed: May 2025.

[150] CSA. Matter - Connectivity Standards Alliance. 2024. Access mode: https://csa-iot.org/all-solutions/matter/. Accessed: May 2025.

[151] SIG Bluetooth. Bluetooth Low Energy. 2024. Access mode: https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/bluetooth-low-energy/. Accessed: May 2025.

[152] Wi-Fi Alliance. 2024. Access mode: https://www.wi-fi.org/. Accessed: May 2025.

[153] Inc. IFTTT. IFTTT Platform. 2024. Access mode: https://ifttt.com/. Accessed: May 2025.

[154] Telegram. Telegram Bot API. 2024. Access mode: https://core.telegram.org/bots/api. Accessed: May 2025.

[155] WeatherAPI.com. Weather API. 2024. Access mode: https://www.weatherapi.com/. Accessed: May 2025.

[156] IETF. The WebSocket Protocol. 2024. Access mode: https://datatracker.ietf.org/doc/html/rfc6455. RFC 6455, Accessed: May 2025.

[157] Fielding Roy T. RESTful API Design. 2024. Access mode: https://restfulapi.net/. Accessed: May 2025.

[158] Society Internet. TCP/IP Protocol Suite. 2024. Access mode: https://www.internetsociety.org/tutorials/tcp-ip/. Accessed: May 2025.

[159] Systems Espressif. ESP32 Technical Reference Manual. 2024. Access mode: https://www.espressif.com/en/products/socs/esp32. Accessed: May 2025.

[160] IETF. TLS 1.3 - Transport Layer Security. 2024. Access mode: https://datatracker.ietf.org/doc/html/rfc8446. Accessed: May 2025.

[161] Wiki Proxmox. USB passthrough in Proxmox VE. 2024. Access mode: https://pve.proxmox.com/wiki/USB_Device_Passthrough. Accessed: May 2025.

[162] ONVIF. ONVIF: Open Network Video Interface Forum. 2024. Access mode: https://www.onvif.org/. Accessed: May 2025.

[163] Project Frigate. Frigate NVR - Real-time object detection. 2024. Access mode: https://frigate.video/. Accessed: May 2025.

[164] Authors Prometheus. Prometheus Monitoring. 2024. Access mode: https://prometheus.io/. Accessed: May 2025.

[165] Labs Grafana. Grafana: The open observability platform. 2024. Access mode: https://grafana.com/. Accessed: May 2025.

Додаток А

Список публікацій здобувача за темою дисертації та відомості про апробацію результатів дисертації

А.1. Список публікацій здобувача, у яких опубліковані основні наукові результати дисертації:

1. Khomenko, Y., & Babichev, S. (2025). Modular IoT Architecture for Monitoring and Control of Office Environments Based on Home Assistant. IoT, 6(4), 69. https://doi.org/10.3390/iot6040069 (Scopus, WoS, quartile Q1)
2. Babichev, S., Yarema, O., Khomenko, Y., Senchyshen, D., & Durnyak, B. (2025). Sensor-Oriented Framework for Underwater Acoustic Signal Classification Using EMD–Wavelet Filtering and Bayesian-Optimized Random Forest. Sensors, 25(17), 5336. https://doi.org/10.3390/s25175336 (Scopus, WoS, quartile Q1)
3. Khomenko, Y., Babichev, S. (2025). Contemporary Methods, Models, and Software for Implementation and Optimization of IoT (Internet of Things) Systems: A Review. In: Babichev, S., Lytvynenko, V. (eds) Lecture Notes in Data Engineering, Computational Intelligence, and Decision-Making, Volume 2. ISDMCI 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 244. Springer, Cham , pp. 134–158. https://doi.org/10.1007/978-3-031-88483-2_7 (Scopus, book chapter, quartile Q4)

А.2. Список публікацій здобувача, які додатково відображають наукові результати дисертації:

4. Хоменко Є.В., Бабічев С.А. (2025). Автономна IoT-система моніторингу мікроклімату аудиторій на основі відкритої DIY-архітектури. Том 8 № 1 (2025): Прикладні питання математичного моделювання, 2025, 245-254. https://doi.org/10.32782/mathematical-modelling/2025-8-1-24 (Категорія Б)
5. Хоменко Є. (2024).Сучасні методи, моделі та програмні засоби реалізації та оптимізації систем IoT (INTERNET OF THINGS),Збірник наукових праць "Information Technologies in Education"(ITE),(55), 72–84. https://doi.org/10.14308/ite000782 (Категорія Б)

А.3. Список публікацій здобувача, які засвідчують апробацію матеріалів дисертації:

6. Khomenko Ye., Babichev S.(2024). Advantages and disadvantages of modern Internet of Things systems. Матеріали XX Міжнародної наукової інтернет-конференції „Intellectual

Systems for Decision Making and Problems of Computational Intelligence", 20-23 червня 2024 року. С. 41-44. м. Усті на Лабі, Чехія

7. Khomenko Y.(2025). Comparison of commercial and local IoT systems for environmental monitoring in educational institutions. Abstracts of XIX International Scientific and Practical Conference. Krakow, Poland. Pp. 228-234.
8. Senchyshen D., Khomenko Y. (2025). Integration of IoT sensors into the KSU24 digital learning environment: infrastructure and pedagogical implications. Abstracts of XX International Scientific and Practical Conference. Plovdiv, Bulgaria. Pp. 222-226.
9. Khomenko Y. (2025) Modern voice assistants of intellectual systems and their use in Internet of Things (IoT) systems. Abstracts of XXII International Scientific and Practical Conference. Krakow, Poland. Pp. 208-214.
10. Хоменко Є., Лемещук О. (2023). Технічні аспекти та норми для систем керування «розумний будинок» в умовах війни, Матеріали IV International Scientific and Theoretical Conference „Features of the development of modern science in the pandemic's era,May 19, 2023. P. 73–75. Berlin, Germany.

Додаток Б

Програмний код реалізації блоку «Climate»

Б.1. Код сторінки «Climate»

```
- title: Climate
cards: []
type: sections
sections:
- type: grid
cards:
- type: heading
icon: mdi:weather-hazy
heading: Weather
heading_style: title
- type: custom:weather-card
entity: weather.forecast_golovna
current: true
details: true
forecast: true
hourly_forecast: true
- type: custom:horizon-card
- show_current: true
show_forecast: true
type: weather-forecast
entity: weather.forecast_golovna
forecast_type: hourly
name: Ivano-Frankivsk
secondary_info_attribute: pressure
- show_current: false
show_forecast: true
type: weather-forecast
entity: weather.forecast_golovna
forecast_type: daily
secondary_info_attribute: temperature
name: Hourly
column_span: 1
- type: grid
cards:
- type: heading
icon: mdi:air-filter
heading: Air quality information
heading_style: title
- type: entities
entities:
- entity: sensor.lun_misto_air_298_pm1
name: Parts PM1
- entity: sensor.lun_misto_air_298_pm2_5
name: Parts PM 2,5
- entity: sensor.lun_misto_air_298_pm10
name: Parts PM 10
title: Air pollution in the city
- type: entities
entities:
```

```
- entity: sensor.aqara_air_quality_air_quality
name: Overall air quality
- entity: sensor.aqara_air_quality_voc
name: Overall score VOC
title: Air quality and VOC in house
- type: grid
cards:
- type: heading
icon: mdi:home-thermometer
heading: Room indicators
heading_style: title
grid_options:
columns: full
rows: 1
- type: vertical-stack
cards:
- type: custom:mini-graph-card
icon: mdi:temperature-celsius
name: Average temperature
entities:
- entity: sensor.room_median_temperature
- type: horizontal-stack
cards:
- type: custom:mini-graph-card
entities:
- sensor.nous_temperature_sensor_temperature
line_color: '#ff513a'
line_width: 8
name: Room 1
- type: custom:mini-graph-card
entities:
- sensor.xiaomi_temperature_sensor_temperature
line_color: orange
line_width: 8
name: Room 2
- type: grid
cards:
- type: heading
icon: mdi:fridge
heading: Climate control devices
heading_style: title
- type: humidifier
entity: humidifier.uvlazhnitel
features:
- type: humidifier-toggle
- type: humidifier-modes
style: dropdown
show_current_as_primary: false
name: Air humidifier
grid_options:
columns: 12
rows: 4
- type: entities
entities:
- entity: binary_sensor.uvlazhnitel_water_tank
name: Humidifier water tank
- entity: sensor.uvlazhnitel_water_level
name: Humidifier water level
title: Humidifier parameters
- type: thermostat
entity: climate.zhimi_heater_zb1_7c_c2_94_f5_40_28
```

```
name: Heater
max_columns: 4
header: {}
badges: []
```

Б.2. Автоматизації розділу «Climate»

Для забезпечення автономної роботи кліматичних пристроїв та підтримки оптимальних показників мікроклімату в системі розумного будинку було розроблено низку автоматизацій. Нижче наведено конфігураційний код сценаріїв.

Автоматизація 1. Автоматичне підтримання температурного режиму.

```
alias: "Climate: Automatic room heating"
description: "Turns on the heater if the median temperature drops below 20C, and turns it off at 22C."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.room_median_temperature
below: 20
id: "too_cold"
- platform: numeric_state
entity_id: sensor.room_median_temperature
above: 22
id: "warm_enough"
condition: []
action:
- choose:
- conditions:
- condition: trigger
id: "too_cold"
sequence:
- service: climate.set_hvac_mode
target:
entity_id: climate.zhimi_heater_zb1_7c_c2_94_f5_40_28
data:
hvac_mode: heat
- service: climate.set_temperature
target:
entity_id: climate.zhimi_heater_zb1_7c_c2_94_f5_40_28
data:
temperature: 23
- conditions:
- condition: trigger
id: "warm_enough"
sequence:
- service: climate.turn_off
target:
entity_id: climate.zhimi_heater_zb1_7c_c2_94_f5_40_28
```

Автоматизація 2. Інтелектуальне сповіщення щодо провітрювання.

```
alias: "Climate: Smart ventilation alert"
description: "Checks indoor VOC and outdoor PM2.5 to suggest window ventilation."
trigger:
- platform: numeric_state
entity_id: sensor.aqara_air_quality_voc
above: 300
```

```
action:
- choose:
- conditions:
- condition: numeric_state
entity_id: sensor.lun_misto_air_298_pm2_5
below: 25
sequence:
- service: notify.notify
data:
title: "Time to ventilate"
message: "Indoor VOC level is high. Outdoor air is clean, you can open the windows."
- conditions:
- condition: numeric_state
entity_id: sensor.lun_misto_air_298_pm2_5
above: 25
sequence:
- service: notify.notify
data:
title: "Poor air quality alert"
message: "Indoor VOC is high, but outdoor PM2.5 is also elevated. Better keep windows closed."
```

Автоматизація 3. Контроль наявності води у зволожувачі.

```
alias: "Climate: Humidifier out of water"
description: "Turns off the humidifier when the water tank is empty and sends a notification."
trigger:
- platform: state
entity_id: binary_sensor.uvlazhnitel_water_tank
to: "on"
- platform: numeric_state
entity_id: sensor.uvlazhnitel_water_level
below: 5
action:
- service: humidifier.turn_off
target:
entity_id: humidifier.uvlazhnitel
- service: notify.notify
data:
title: "Humidifier is empty"
message: "Water tank is empty. Humidifier turned off, please refill."
```

Автоматизація 4. Компенсація втрати вологості під час обігріву.

```
alias: "Climate: Auto humidify while heating"
description: "Turns on the humidifier when the heater is active and humidity is below 45 percent, turns off at 55
   percent."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.nous_temperature_sensor_humidity
below: 45
id: "low_humidity"
- platform: numeric_state
entity_id: sensor.nous_temperature_sensor_humidity
above: 55
id: "high_humidity"
- platform: state
entity_id: climate.zhimi_heater_zb1_7c_c2_94_f5_40_28
to: "heat"
id: "heater_on"
```

```
condition: []
action:
- choose:
- conditions:
- condition: or
conditions:
- condition: trigger
id: "low_humidity"
- condition: trigger
id: "heater_on"
- condition: state
entity_id: climate.zhimi_heater_zb1_7c_c2_94_f5_40_28
state: "heat"
- condition: numeric_state
entity_id: sensor.nous_temperature_sensor_humidity
below: 55
- condition: state
entity_id: binary_sensor.uvlazhnitel_water_tank
state: "off"
sequence:
- service: humidifier.turn_on
target:
entity_id: humidifier.uvlazhnitel
- conditions:
- condition: trigger
id: "high_humidity"
sequence:
- service: humidifier.turn_off
target:
entity_id: humidifier.uvlazhnitel
```

Автоматизація 5. Запобігання утворенню конденсату на вікнах.

```
alias: "Climate: Window condensation prevention"
description: "Adjusts target humidity based on outdoor temperature from weather integration."
mode: restart
trigger:
- platform: state
entity_id: weather.forecast_golovna
attribute: temperature
action:
- choose:
- conditions:
- condition: numeric_state
entity_id: weather.forecast_golovna
attribute: temperature
below: -5
sequence:
- service: humidifier.set_humidity
target:
entity_id: humidifier.uvlazhnitel
data:
humidity: 40
- conditions:
- condition: numeric_state
entity_id: weather.forecast_golovna
attribute: temperature
above: -5
sequence:
- service: humidifier.set_humidity
target:
```

```
entity_id: humidifier.uvlazhnitel
data:
humidity: 50
```

Автоматизація 6. Розклад нічного кліматичного режиму.

```
alias: "Climate: Night mode schedule"
description: "Lowers heater temperature at night for better sleep and restores it in the morning."
trigger:
- platform: time
at: "23:00:00"
id: "night"
- platform: time
at: "07:00:00"
id: "morning"
action:
- choose:
- conditions:
- condition: trigger
id: "night"
sequence:
- service: climate.set_temperature
target:
entity_id: climate.zhimi_heater_zb1_7c_c2_94_f5_40_28
data:
temperature: 19
- conditions:
- condition: trigger
id: "morning"
sequence:
- service: climate.set_temperature
target:
entity_id: climate.zhimi_heater_zb1_7c_c2_94_f5_40_28
data:
temperature: 22
```

Автоматизація 7. Ранкове інформаційне зведення.

```
alias: "Climate: Morning status report"
description: "Sends a daily notification with indoor and outdoor air quality."
trigger:
- platform: time
at: "08:00:00"
action:
- service: notify.notify
data:
title: "Morning Climate Report"
message: >
Indoor Temp: {{ states('sensor.room_median_temperature') }}C.
Indoor VOC: {{ states('sensor.aqara_air_quality_voc') }}.
Outdoor PM2.5: {{ states('sensor.lun_misto_air_298_pm2_5') }}.
Weather: {{ states('weather.forecast_golovna') }}.
```

Автоматизація 8. Критичний рівень забруднення зовнішнього повітря.

```
alias: "Climate: Severe outdoor pollution alert"
description: "Sends an alert if outdoor PM2.5 or PM10 reaches dangerous levels."
trigger:
- platform: numeric_state
entity_id: sensor.lun_misto_air_298_pm2_5
```

```
above: 50
- platform: numeric_state
entity_id: sensor.lun_misto_air_298_pm10
above: 100
action:
- service: notify.notify
data:
title: "Severe Air Pollution"
message: "Outdoor air quality has dropped significantly. Keep windows closed."
```

Додаток В

Програмний код реалізації блоку «Secure»

В.1. Код сторінки «Secure»

```
- title: Secure
path: security
icon: ''
cards: []
sections:
- type: grid
cards:
- type: heading
icon: mdi:security
heading_style: title
heading: Danger
- type: custom:alarmo-card
entity: alarm_control_panel.alarmo
name: Secure system
keep_keypad_visible: false
use_clear_icon: true
button_scale_actions: 1.1
button_scale_keypad: 1.1
states: {}
show_messages: true
show_ready_indicator: true
show_bypassed_sensors: true
use_code_dialog: false
- type: entities
entities:
- entity: binary_sensor.ivano_frankivska_oblast_air
name: Air alarm
- entity: binary_sensor.ivano_frankivska_oblast_artillery
name: Artillery alarm
- entity: binary_sensor.ivano_frankivska_oblast_chemical
name: Chemical alarm
- entity: binary_sensor.ivano_frankivska_oblast_nuclear
name: Nuclear alarm
- entity: binary_sensor.ivano_frankivska_oblast_urban_fights
name: Urban fights alarm
title: Alarm in Ivano-Frankivsk
layout_options:
grid_columns: 4
grid_rows: 5
- type: vertical-stack
cards:
- type: entities
entities:
- entity: binary_sensor.meian_water_sensor_water_leak
name: Room 1
- entity: binary_sensor.meian_water_sensor_tamper
name: Device tampering
- entity: binary_sensor.tuya_water_leak_water_leak
```

```
name: Room 2
- entity: binary_sensor.tuya_water_leak_tamper
name: Device tampering
title: Water leak sensors
- type: entities
entities:
- entity: binary_sensor.dygsm_gas_sensor_gas
name: 'Gas leak '
secondary_info: none
icon: mdi:gas-burner
- entity: sensor.dygsm_gas_sensor_gas_value
name: 'Gas concentration '
icon: mdi:gauge-empty
- entity: sensor.dygsm_gas_sensor_self_test_result
icon: mdi:check-circle-outline
name: Device self-test result
- entity: binary_sensor.tuya_smoke_sensor_smoke
name: Presence of smoke
icon: mdi:smoke
- entity: binary_sensor.tuya_smoke_sensor_tamper
name: Device tampering
icon: mdi:dishwasher-alert
title: Gas in house
- type: grid
cards:
- type: heading
icon: mdi:motion-sensor
heading: 'Motion '
heading_style: title
- show_name: true
show_icon: true
show_state: true
type: glance
entities:
- entity: person.Person_1
name: Person 1
- entity: person.Person_2
name: Person 2
title: Persons
- type: history-graph
entities:
- entity: person.Person_1
name: Person 1
- entity: device_tracker.laptop
name: Person 2
title: Persons location history
- type: vertical-stack
cards:
- type: entities
entities:
- entity: binary_sensor.aqara_move_sensor_occupancy
name: Motion zone 1
icon: ''
- type: history-graph
entities:
- entity: binary_sensor.aqara_move_sensor_occupancy
name: History zone 1
hours_to_show: 6
- type: entities
entities:
- entity: binary_sensor.tuya_move_sensor_occupancy
```

```
name: Motion zone 2
icon: mdi:motion-sensor-off
- type: history-graph
entities:
- entity: binary_sensor.tuya_move_sensor_occupancy
name: History zone 2
hours_to_show: 6
- type: entities
entities:
- entity: binary_sensor.xiaomi_move_sensor_occupancy
name: Motion zone 3
icon: mdi:motion-sensor-off
- type: history-graph
entities:
- entity: binary_sensor.xiaomi_move_sensor_occupancy
name: History zone 3
icon: mdi:motion-sensor-off
hours_to_show: 6
- type: glance
entities:
- entity: binary_sensor.tuya_human_breathe_sensor_presence
name: Breathe detection
- entity: sensor.tuya_human_breathe_sensor_illuminance
name: Illuminance
- entity: sensor.tuya_human_breathe_sensor_target_distance
name: Distance to target
grid_options:
columns: 12
rows: 8
- type: grid
cards:
- type: heading
icon: mdi:cctv
heading: Secure Camera and House
heading_style: title
- show_state: true
show_name: true
camera_view: live
fit_mode: cover
type: picture-entity
entity: camera.moes_camera_profile_1
camera_image: camera.moes_camera_profile_1
name: Camera Onvif
- show_name: true
show_icon: true
show_state: true
type: glance
entities:
- entity: binary_sensor.moes_camera_cell_motion_detection
name: Motion
- entity: switch.moes_camera_autofocus
name: Autofocus
- entity: switch.moes_camera_ir_lamp
name: IR-Lamp
- entity: button.moes_camera_reboot
name: Reboot
title: ONVIF camera settings
- type: history-graph
entities:
- entity: binary_sensor.moes_camera_cell_motion_detection
title: Motion zone secure camera
```

```
hours_to_show: 8
- type: vertical-stack
cards:
- type: entities
entities:
- entity: binary_sensor.all_doors_and_windows_closed
- entity: binary_sensor.tuya_door_sensor_contact
name: Main door
icon: mdi:door-closed-lock
- entity: binary_sensor.tuya_door_sensor_tamper
name: Device tampering
icon: mdi:dishwasher-alert
- entity: binary_sensor.xiaomi_door_sensor_contact
name: Window room 1
icon: mdi:window-closed-variant
- entity: binary_sensor.xiaomi_door_sensor_ble_opening
name: Window room2
icon: mdi:window-closed-variant
title: Windows and doors
type: sections
max_columns: 3
```

В.2. Автоматизації розділу «Secure»

Для забезпечення комплексного моніторингу безпеки та оперативного реагування на інциденти розроблено програмні сценарії, які об'єднують датчики руху, контролю периметра, камери спостереження та системи виявлення техногенних аварій.

Автоматизація 1. Критичне сповіщення про техногенну небезпеку.

```
alias: "Security: Critical hazard detected"
description: "Triggers immediate notification when water leak, gas leak, or smoke is detected."
mode: single
trigger:
- platform: state
entity_id: binary_sensor.meian_water_sensor_water_leak
to: "on"
- platform: state
entity_id: binary_sensor.tuya_water_leak_water_leak
to: "on"
- platform: state
entity_id: binary_sensor.dygsm_gas_sensor_gas
to: "on"
- platform: state
entity_id: binary_sensor.tuya_smoke_sensor_smoke
to: "on"
action:
- service: notify.notify
data:
title: "CRITICAL: Hazard Detected"
message: "Water leak, gas leak, or smoke sensor has been triggered. Immediate action required."
```

Автоматизація 2. Автоматичне встановлення системи під охорону.

```
alias: "Security: Auto-arm system when away"
description: "Automatically arms the Alarmo panel when all residents leave the premises."
mode: single
```

```
trigger:
- platform: state
entity_id: person.Person_1
to: "not_home"
- platform: state
entity_id: person.Person_2
to: "not_home"
condition:
- condition: state
entity_id: person.Person_1
state: "not_home"
- condition: state
entity_id: person.Person_2
state: "not_home"
action:
- service: alarm_control_panel.alarm_arm_away
target:
entity_id: alarm_control_panel.alarmo
```

Автоматизація 3. Контроль цілісності периметра під час охорони.

```
alias: "Security: Open perimeter warning"
description: "Sends an alert if the alarm system is armed while any door or window remains open."
mode: single
trigger:
- platform: state
entity_id: alarm_control_panel.alarmo
to: "armed_away"
condition:
- condition: state
entity_id: binary_sensor.all_doors_and_windows_closed
state: "off"
action:
- service: notify.notify
data:
title: "Security Warning: Perimeter Open"
message: "The security system is armed, but one or more doors/windows are currently open."
```

Автоматизація 4. Фіксація апаратного втручання (Tamper Alert).

```
alias: "Security: Device tampering detected"
description: "Notifies the user if unauthorized physical access to any security sensor is detected."
mode: queued
trigger:
- platform: state
entity_id: binary_sensor.meian_water_sensor_tamper
to: "on"
- platform: state
entity_id: binary_sensor.tuya_water_leak_tamper
to: "on"
- platform: state
entity_id: binary_sensor.tuya_smoke_sensor_tamper
to: "on"
- platform: state
entity_id: binary_sensor.tuya_door_sensor_tamper
to: "on"
action:
- service: notify.notify
data:
title: "Security Alert: Sensor Tampering"
message: "A security device has reported a tamper event. Please inspect the hardware."
```

Автоматизація 5. Інтеграція сповіщень цивільного захисту.

```
alias: "Security: Civil defense alert"
description: "Broadcasts an urgent notification upon receiving an air raid or critical threat signal."
mode: parallel
trigger:
- platform: state
entity_id: binary_sensor.ivano_frankivska_oblast_air
to: "on"
- platform: state
entity_id: binary_sensor.ivano_frankivska_oblast_nuclear
to: "on"
action:
- service: notify.notify
data:
title: "Civil Defense Alert"
message: "Emergency alert declared in your region. Please proceed to safety protocols."
```

Автоматизація 6. Інтелектуальний запис відео за детекцією руху.

```
alias: "Security: Camera motion capture"
description: "Records a short video clip if the ONVIF camera detects motion while the system is armed."
mode: single
trigger:
- platform: state
entity_id: binary_sensor.moes_camera_cell_motion_detection
to: "on"
condition:
- condition: state
entity_id: alarm_control_panel.alarmo
state: "armed_away"
action:
- service: camera.record
target:
entity_id: camera.moes_camera_profile_1
data:
duration: 15
filename: "/media/security_capture_{{ now().strftime('%Y%m%d_%H%M%S') }}.mp4"
```

Автоматизація 7. Автоматичне зняття системи з охорони.

```
alias: "Security: Auto-disarm on arrival"
description: "Disarms the Alarmo panel when any resident enters the home zone or connects to the local network."
mode: single
trigger:
- platform: state
entity_id: person.Person_1
to: "home"
- platform: state
entity_id: person.Person_2
to: "home"
action:
- service: alarm_control_panel.alarm_disarm
target:
entity_id: alarm_control_panel.alarmo
```

Автоматизація 8. Інтелектуальне керування інфрачервоною підсвіткою камери.

```
alias: "Security: Smart camera IR control"
description: "Enables camera IR lamp when room illuminance drops below threshold, utilizing the Tuya breathe sensor
   data."
```

```
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.tuya_human_breathe_sensor_illuminance
below: 15
action:
- service: switch.turn_on
target:
entity_id: switch.moes_camera_ir_lamp
```

Автоматизація 9. Високоточна детекція вторгнення .

```
alias: "Security: Radar intrusion detection"
description: "Triggers the alarm if the mmWave radar detects human presence (breathing) while the system is armed."
mode: single
trigger:
- platform: state
entity_id: binary_sensor.tuya_human_breathe_sensor_presence
to: "on"
condition:
- condition: state
entity_id: alarm_control_panel.alarmo
state: "armed_away"
action:
- service: alarm_control_panel.alarm_trigger
target:
entity_id: alarm_control_panel.alarmo
```

Автоматизація 10. Автоматизоване обслуговування підсистеми відеонагляду.

```
alias: "Security: Auto-reboot camera on failure"
description: "Reboots the ONVIF camera using the device hardware button entity if the video stream becomes unavailable
   ."
mode: single
trigger:
- platform: state
entity_id: camera.moes_camera_profile_1
to: "unavailable"
for:
minutes: 2
action:
- service: button.press
target:
entity_id: button.moes_camera_reboot
- service: notify.notify
data:
title: "System Maintenance Action"
message: "The ONVIF camera went offline and was automatically rebooted to restore the security feed."
```

Автоматизація 11. Автоматичне вимкнення інфрачервоної підсвітки.

```
alias: "Security: Smart camera IR off"
description: "Disables camera IR lamp when room illuminance rises above threshold, restoring color vision."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.tuya_human_breathe_sensor_illuminance
above: 50
action:
- service: switch.turn_off
```

```
target:
entity_id: switch.moes_camera_ir_lamp
```

Додаток Г

Програмний код реалізації блоку «Light»

Г.1. Код сторінки «Light»

```
- title: Light
path: light
cards: []
type: sections
sections:
- type: grid
cards:
- type: heading
icon: mdi:lightbulb-on-outline
heading: Light monitoring
heading_style: title
- type: custom:horizon-card
- type: entities
entities:
- entity: sensor.golovna_sun_civil_daylight
name: Civil daylight
- entity: sensor.golovna_sun_civil_night
name: Civil night
- type: custom:mini-graph-card
icon: mdi:server
name: Civil changes
entities:
- entity: sensor.golovna_sun_civil_daylight
- sensor.golovna_sun_civil_night
- type: light
entity: light.light_global_room1
name: Global Light
icon: mdi:lightbulb-group
- show_name: true
show_icon: true
type: button
entity: switch.sonoff_100168396a_1
grid_options:
columns: 12
rows: 2
show_state: true
name: Adaptive Light
icon: mdi:lightbulb-auto
- type: grid
cards:
- type: heading
heading: Room 1 Light
heading_style: title
- type: vertical-stack
cards:
- show_name: true
show_icon: true
show_state: true
```

```
type: glance
entities:
- entity: light.lidl_light_e27
name: Main Light
- entity: switch.tuya_plug
name: Ambient light
title: Light bulb room 1
- type: custom:mushroom-light-card
entity: light.tuya_led_strip_1
name: Backlight room 1
use_light_color: true
show_brightness_control: true
show_color_control: true
show_color_temp_control: false
fill_container: false
collapsible_controls: true
- type: custom:mushroom-light-card
entity: light.tuya_led_strip_2
name: Backlight room 2
use_light_color: true
show_brightness_control: true
show_color_control: true
show_color_temp_control: false
fill_container: false
collapsible_controls: true
- type: entities
entities:
- entity: sensor.xiaomi_light_sensor_illuminance
- entity: binary_sensor.xiaomi_door_sensor_ble_light
name: Auditorium lighting level 3
- type: grid
cards:
- type: heading
heading: Room 2 Light
heading_style: title
- type: vertical-stack
cards:
- show_name: true
show_icon: true
show_state: true
type: glance
entities:
- entity: light.lidl_light_e27
name: Main Light
- entity: light.lidl_light_e14
name: Ambient light
title: Light bulb room 2
- type: custom:mushroom-light-card
entity: light.tuya_led_strip_2
name: Backlight room 1
use_light_color: true
show_brightness_control: true
show_color_control: true
show_color_temp_control: false
fill_container: false
collapsible_controls: true
- type: custom:mushroom-light-card
entity: light.tuya_led_strip_2
name: Backlight room 2
use_light_color: true
show_brightness_control: true
```

```
show_color_control: true
show_color_temp_control: false
fill_container: false
collapsible_controls: true
- type: entities
entities:
- entity: sensor.xiaomi_light_sensor_illuminance
- entity: binary_sensor.xiaomi_door_sensor_ble_light
name: Auditorium lighting level 3
- show_name: true
show_icon: true
type: button
entity: switch.sonoff_100168396a_1
grid_options:
columns: 12
rows: 2
show_state: true
name: Desk Light
- type: grid
cards:
- type: heading
heading: Daylight Management
heading_style: title
- type: vertical-stack
cards:
- type: entities
entities:
- entity: sensor.xiaomi_light_sensor_illuminance
name: Window 1 Light Sensor
- type: custom:mushroom-cover-card
entity: cover.tuya_roller_blind_1
name: Window 1 Blinds
show_position_control: true
show_tilt_position_control: false
show_buttons_control: true
fill_container: true
grid_options:
columns: 12
rows: 2
- type: vertical-stack
cards:
- type: entities
entities:
- entity: sensor.xiaomi_light_sensor_illuminance
name: Window 2 Light Sensor
- type: custom:mushroom-cover-card
entity: cover.tuya_roller_blind_2
name: Window 2 Blinds
show_position_control: true
show_tilt_position_control: false
show_buttons_control: true
fill_container: true
grid_options:
columns: 12
rows: 2
- type: vertical-stack
cards:
- type: entities
entities:
- entity: sensor.xiaomi_light_sensor_illuminance
name: Window 3 Light Sensor
```

```
- type: custom:mushroom-cover-card
entity: cover.tuya_roller_blind_3
name: Window 3 Blinds
show_position_control: true
show_tilt_position_control: false
show_buttons_control: true
fill_container: true
grid_options:
columns: 12
rows: 2
```

Г.2. Автоматизації розділу «Light»

Підсистема освітлення та управління природним світлом реалізує концепцію динамічного адаптивного освітлення (Daylight Harvesting). Розроблені сценарії забезпечують синхронізацію роботи рулонних штор із показниками освітленості та астрономічними подіями, а також автоматичне балансування штучних джерел світла.

Автоматизація 1. Автоматичне закриття штор у нічний час.

```
alias: "Lighting: Automated night blinds"
description: "Closes all roller blinds when civil night begins to ensure privacy and thermal insulation."
mode: single
trigger:
- platform: state
entity_id: sensor.golovna_sun_civil_night
to: "on"
action:
- service: cover.close_cover
target:
entity_id:
- cover.tuya_roller_blind_1
- cover.tuya_roller_blind_2
- cover.tuya_roller_blind_3
```

Автоматизація 2. Динамічне керування природним освітленням (Daylight Management).

```
alias: "Lighting: Dynamic shading based on illuminance"
description: "Partially closes the blinds to prevent glare and overheating when direct sunlight causes excessive room
   illuminance."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.xiaomi_light_sensor_illuminance
above: 3000
for:
minutes: 5
action:
- service: cover.set_cover_position
target:
entity_id:
- cover.tuya_roller_blind_1
- cover.tuya_roller_blind_2
- cover.tuya_roller_blind_3
data:
position: 30
```

Автоматизація 3. Адаптивне робоче освітлення.

```
alias: "Lighting: Workspace illumination"
description: "Turns on the Sonoff desk light automatically if the room illuminance drops below a comfortable threshold
   ."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.xiaomi_light_sensor_illuminance
below: 300
for:
minutes: 2
condition:
- condition: state
entity_id: sensor.golovna_sun_civil_daylight
state: "on"
action:
- service: switch.turn_on
target:
entity_id: switch.sonoff_100168396a_1
```

Автоматизація 4. Увімкнення вечірнього фонового освітлення.

```
alias: "Lighting: Evening ambient lights"
description: "Activates LED strips and ambient plugs at a dim level when daylight ends to provide comfortable
   transitional lighting."
mode: single
trigger:
- platform: state
entity_id: sensor.golovna_sun_civil_daylight
to: "off"
action:
- service: light.turn_on
target:
entity_id:
- light.tuya_led_strip_1
- light.tuya_led_strip_2
data:
brightness_pct: 40
color_name: "warm white"
- service: switch.turn_on
target:
entity_id: switch.tuya_plug
```

Автоматизація 5. Синхронізація глобального освітлення кімнати.

```
alias: "Lighting: Global room off"
description: "Turns off all lighting entities in the room simultaneously when the global light group is triggered off
   ."
mode: single
trigger:
- platform: state
entity_id: light.light_global_room1
to: "off"
action:
- service: light.turn_off
target:
entity_id:
- light.lidl_light_e27
- light.lidl_light_e14
- light.tuya_led_strip_1
```

```
- light.tuya_led_strip_2
- service: switch.turn_off
target:
entity_id:
- switch.sonoff_100168396a_1
- switch.tuya_plug
```

Автоматизація 6. Ранкове розкриття штор.

```
alias: "Lighting: Morning blinds opening"
description: "Opens all roller blinds when civil daylight starts, allowing natural light into the premises."
mode: single
trigger:
- platform: state
entity_id: sensor.golovna_sun_civil_daylight
to: "on"
action:
- service: cover.open_cover
target:
entity_id:
- cover.tuya_roller_blind_1
- cover.tuya_roller_blind_2
- cover.tuya_roller_blind_3
```

Автоматизація 7. Глобальне вимкнення освітлення на основі геозонування.

```
alias: "Lighting: All off when leaving home"
description: "Turns off all lights and smart plugs when all residents leave the home zone to optimize energy
   consumption."
mode: single
trigger:
- platform: state
entity_id: person.Person_1
to: "not_home"
- platform: state
entity_id: person.Person_2
to: "not_home"
condition:
- condition: state
entity_id: person.Person_1
state: "not_home"
- condition: state
entity_id: person.Person_2
state: "not_home"
action:
- service: light.turn_off
target:
entity_id:
- light.light_global_room1
- light.tuya_led_strip_1
- light.tuya_led_strip_2
- service: switch.turn_off
target:
entity_id:
- switch.sonoff_100168396a_1
- switch.tuya_plug
```

Автоматизація 8. Режим «Master Sleep».

```
alias: "Lighting: Sleep mode activation"
```

```
description: "Turns off all lighting, closes all blinds, and prepares the house for the night mode via a single button
    press."
mode: single
trigger:
- platform: state
entity_id: input_button.sleep_mode
action:
- service: light.turn_off
target:
entity_id:
- light.light_global_room1
- light.lidl_light_e27
- light.lidl_light_e14
- light.tuya_led_strip_1
- light.tuya_led_strip_2
- service: cover.close_cover
target:
entity_id:
- cover.tuya_roller_blind_1
- cover.tuya_roller_blind_2
- cover.tuya_roller_blind_3
- service: switch.turn_off
target:
entity_id:
- switch.sonoff_100168396a_1
- switch.tuya_plug
```

Автоматизація 9. Світлова сцена «Movie Mode».

```
alias: "Lighting: Movie mode scene"
description: "Dims the room for media consumption: turns off main lights, closes blinds, and sets LED strips to a dim
   blue color."
mode: single
trigger:
- platform: state
entity_id: input_button.movie_mode
action:
- service: light.turn_off
target:
entity_id: light.light_global_room1
- service: cover.close_cover
target:
entity_id:
- cover.tuya_roller_blind_1
- cover.tuya_roller_blind_2
- cover.tuya_roller_blind_3
- service: light.turn_on
target:
entity_id:
- light.tuya_led_strip_1
- light.tuya_led_strip_2
data:
brightness_pct: 10
color_name: "blue"
```

Автоматизація 10. Імітація світанку (Wake-up Light).

```
alias: "Lighting: Sunrise simulation wake-up"
description: "Gradually increases the brightness of the LED strips over 15 minutes before the morning alarm to
   simulate a natural sunrise."
```

```
mode: single
trigger:
- platform: time
at: "06:45:00"
condition:
- condition: time
weekday:
- mon
- tue
- wed
- thu
- fri
action:
- service: light.turn_on
target:
entity_id:
- light.tuya_led_strip_1
- light.tuya_led_strip_2
data:
brightness_pct: 1
color_name: "warm white"
- delay:
hours: 0
minutes: 1
seconds: 0
milliseconds: 0
- service: light.turn_on
target:
entity_id:
- light.tuya_led_strip_1
- light.tuya_led_strip_2
data:
brightness_pct: 100
transition: 900
```

Додаток Д

Програмний код реалізації блоку «System»

Д.1. Код сторінки «System»

```
- title: System
path: system
cards: []
type: sections
sections:
- type: grid
cards:
- type: heading
icon: mdi:fridge
heading: System indicators
heading_style: title
- type: entities
entities:
- entity: sensor.date
name: Data
- entity: sensor.time
name: Time
- entity: sensor.uptime
name: Uptime
- entity: sensor.archer_c54_ac1200_dual_band_wi_fi_router_external_ip
name: Current IP
- entity: sensor.cpu_speed
name: CPU frequency
layout_options:
grid_columns: 4
grid_rows: 4
- show_name: true
show_icon: true
show_state: true
type: glance
entities:
- entity: sensor.processor_use
name: CPU load
- entity: sensor.memory_use_percent
name: RAM loading
- entity: sensor.disk_use_percent
name: Space taken
- entity: sensor.disk_free
name: Free space
columns: 2
state_color: false
layout_options:
grid_columns: 4
grid_rows: 4
- type: heading
icon: mdi:fridge
heading: Server power supply
heading_style: title
```

```
- chart_type: line
period: 5minute
type: statistics-graph
entities:
- sensor.myups_input_voltage
stat_types:
- mean
- min
- max
days_to_show: 1
hide_legend: false
logarithmic_scale: true
title: Input voltage
grid_options:
columns: 12
rows: 6
- type: grid
cards:
- type: heading
icon: mdi:fridge
heading: Internet connection
heading_style: title
- type: custom:mini-graph-card
icon: mdi:server
name: Internet connection schedule
entities:
- entity: sensor.speedtest_download
- sensor.speedtest_upload
- type: glance
entities:
- entity: sensor.speedtest_download
name: Download speed
- entity: sensor.speedtest_upload
name: Upload speed
- entity: sensor.speedtest_ping
name: Ping
- type: grid
cards:
- type: heading
icon: mdi:devices
heading: Devices in network
heading_style: title
- type: entities
entities:
- entity: binary_sensor.router
- entity: binary_sensor.proxmox_server
- entity: binary_sensor.development_server
- entity: binary_sensor.homepod_mini
title: Infrastructure
- type: entities
entities:
- entity: binary_sensor.pc
- entity: binary_sensor.laptop
- entity: binary_sensor.phone_1
- entity: binary_sensor.phone_2
title: Client Devices
- type: entities
entities:
- entity: binary_sensor.power_station
- entity: binary_sensor.humidifier
- entity: binary_sensor.heater
```

```
title: Home Appliances
- type: grid
cards:
- type: heading
icon: mdi:fridge
heading: Relevance of system components
heading_style: title
- type: custom:mushroom-update-card
entity: update.home_assistant_supervisor_update
layout_options:
grid_columns: 4
grid_rows: 1
fill_container: false
collapsible_controls: false
show_buttons_control: false
- type: custom:mushroom-update-card
entity: update.home_assistant_core_update
layout_options:
grid_columns: 4
grid_rows: 1
- type: custom:mushroom-update-card
entity: update.home_assistant_operating_system_update
layout_options:
grid_columns: 4
grid_rows: 1
- type: custom:mushroom-update-card
entity: update.mosquitto_broker_update
collapsible_controls: false
layout_options:
grid_columns: 4
grid_rows: 1
- type: custom:mushroom-update-card
entity: update.zigbee2mqtt_update
layout_options:
grid_columns: 4
grid_rows: 1
- type: entities
entities:
- entity: binary_sensor.backups_stale
- entity: sensor.backup_last_successful_automatic_backup
- entity: sensor.backup_backup_manager_state
- entity: sensor.backup_next_scheduled_automatic_backup
title: Backup Status
max_columns: 4
```

Д.2. Автоматизації розділу «System»

Автоматизація системного моніторингу спрямована на забезпечення високої доступності (High Availability) та відмовостійкості апаратно-програмного комплексу. Розроблені сценарії охоплюють контроль використання апаратних ресурсів, моніторинг стану джерел безперебійного живлення, перевірку цілісності резервних копій та відстеження доступності ключових мережевих вузлів.

Автоматизація 1. Моніторинг вичерпання апаратних ресурсів.

```
alias: "System: Critical resource usage alert"
```

```
description: "Triggers a notification if disk space or RAM usage exceeds 90 percent to prevent system crashes."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.disk_use_percent
above: 90
id: "disk_full"
- platform: numeric_state
entity_id: sensor.memory_use_percent
above: 90
id: "ram_full"
action:
- choose:
- conditions:
- condition: trigger
id: "disk_full"
sequence:
- service: notify.notify
data:
title: "System Alert: Storage Critical"
message: "Main storage usage exceeded 90 percent. Please free up disk space."
- conditions:
- condition: trigger
id: "ram_full"
sequence:
- service: notify.notify
data:
title: "System Alert: RAM Critical"
message: "Memory usage exceeded 90 percent. Check running containers and processes."
```

Автоматизація 2. Детекція втрати основного електроживлення (UPS).

```
alias: "System: Power outage detected"
description: "Monitors the UPS input voltage and triggers an alert if the power grid fails, initiating a grace period
   before potential shutdown."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.myups_input_voltage
below: 100
for:
seconds: 30
action:
- service: notify.notify
data:
title: "Infrastructure Warning: Power Loss"
message: "Main power supply lost. Servers are currently running on battery backup."
```

Автоматизація 3. Контроль доступності серверної інфраструктури.

```
alias: "System: Critical node offline"
description: "Alerts the administrator if the Proxmox hypervisor or development server goes offline."
mode: parallel
trigger:
- platform: state
entity_id: binary_sensor.proxmox_server
to: "off"
for:
minutes: 2
id: "proxmox_down"
```

```
- platform: state
entity_id: binary_sensor.development_server
to: "off"
for:
minutes: 2
id: "dev_down"
action:
- service: notify.notify
data:
title: "Infrastructure Alert: Node Offline"
message: "A critical server node has stopped responding on the network. Immediate verification required."
```

Автоматизація 4. Забезпечення надійності резервних копій.

```
alias: "System: Backup failure alert"
description: "Triggers a notification if the automated backup routine fails and backups are marked as stale."
mode: single
trigger:
- platform: state
entity_id: binary_sensor.backups_stale
to: "on"
action:
- service: notify.notify
data:
title: "Data Protection Alert: Stale Backups"
message: "The scheduled automated backup failed. System configurations and states are at risk."
```

Автоматизація 5. Відстеження зміни зовнішньої IP-адреси.

```
alias: "System: External IP address changed"
description: "Monitors the router for WAN IP changes to ensure external access and dynamic DNS services remain
   functional."
mode: single
trigger:
- platform: state
entity_id: sensor.archer_c54_ac1200_dual_band_wi_fi_router_external_ip
condition:
- condition: template
value_template: "{{ trigger.from_state.state != 'unknown' and trigger.from_state.state != 'unavailable' }}"
action:
- service: notify.notify
data:
title: "Network Event: WAN IP Changed"
message: "The external IP address has been updated to {{ states('sensor.
   archer_c54_ac1200_dual_band_wi_fi_router_external_ip') }}."
```

Автоматизація 6. Сповіщення про доступність системних оновлень.

```
alias: "System: Core and OS updates available"
description: "Notifies the administrator when new versions of Home Assistant Core, Supervisor, or OS are published."
mode: single
trigger:
- platform: state
entity_id: update.home_assistant_core_update
to: "on"
- platform: state
entity_id: update.home_assistant_operating_system_update
to: "on"
action:
- service: notify.notify
```

```
data:
title: "System Update Available"
message: "New updates are available for the core system components. Please review the changelog before applying."
```

Автоматизація 7. Моніторинг зниження швидкості інтернет-з'єднання.

```
alias: "System: Low download speed alert"
description: "Triggers a notification if the download speed drops below 50 Mbps, indicating ISP issues or network
   congestion."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.speedtest_download
below: 50
for:
minutes: 15
action:
- service: notify.notify
data:
title: "Network Degradation: Low Download Speed"
message: "Download speed has dropped below the acceptable threshold. Current speed is {{ states('sensor.
   speedtest_download') }} Mbps."
```

Автоматизація 8. Виявлення аномальної затримки мережі (High Ping).

```
alias: "System: High network latency detected"
description: "Alerts the administrator when the network ping exceeds 150 ms, which may affect remote access and IoT
   responsiveness."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.speedtest_ping
above: 150
for:
minutes: 10
action:
- service: notify.notify
data:
title: "Network Degradation: High Latency"
message: "Network ping is unusually high ({{ states('sensor.speedtest_ping') }} ms). This may cause delays in cloud
   integrations."
```

Автоматизація 9. Фіксація нестабільності напруги в електромережі.

```
alias: "System: Grid voltage instability warning"
description: "Monitors the UPS input voltage for dangerous fluctuations (overvoltage or undervoltage) before a
   complete blackout occurs."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.myups_input_voltage
above: 250
id: "overvoltage"
- platform: numeric_state
entity_id: sensor.myups_input_voltage
below: 190
id: "undervoltage"
condition:
- condition: numeric_state
entity_id: sensor.myups_input_voltage
```

```
above: 50
action:
- choose:
- conditions:
- condition: trigger
id: "overvoltage"
sequence:
- service: notify.notify
data:
title: "Power Grid Alert: Overvoltage"
message: "Input voltage is dangerously high ({{ states('sensor.myups_input_voltage') }}V). UPS is compensating."
- conditions:
- condition: trigger
id: "undervoltage"
sequence:
- service: notify.notify
data:
title: "Power Grid Alert: Undervoltage"
message: "Input voltage has dropped significantly ({{ states('sensor.myups_input_voltage') }}V). Brownout condition
  detected."
```

Автоматизація 10. Втрата зв'язку з критичними IoT-пристроями.

```
alias: "System: IoT appliance offline"
description: "Notifies if crucial home appliances like the humidifier or heater drop off the Wi-Fi network, ensuring
  climate control reliability."
mode: parallel
trigger:
- platform: state
entity_id: binary_sensor.humidifier
to: "off"
for:
minutes: 5
id: "humidifier"
- platform: state
entity_id: binary_sensor.heater
to: "off"
for:
minutes: 5
id: "heater"
action:
- service: notify.notify
data:
title: "IoT Connectivity Alert"
message: "A critical climate appliance has disconnected from the local network. Check power or Wi-Fi coverage."
```

Автоматизація 11. Перезавантаження інтеграцій при відновленні інтернету.

```
alias: "System: Network restored actions"
description: "Automatically triggers a refresh or sends a status update when the main router comes back online after
  an outage."
mode: single
trigger:
- platform: state
entity_id: binary_sensor.router
from: "off"
to: "on"
action:
- service: notify.notify
data:
```

```
title: "Network Restored"
message: "Main router is back online. Local network and internet access should be operating normally."
```

Додаток Е

Програмний код реалізації блоку «Battery»

Е.1. Код сторінки «Battery»

```
- title: Battery
path: battery
cards: []
badges: []
type: sections
sections:
- type: grid
cards:
- type: heading
icon: mdi:transmission-tower
heading: Electrical information
heading_style: title
- title: Energy used today
type: energy-distribution
link_dashboard: true
- type: history-graph
entities:
- entity: sensor.myups_input_voltage
name: Input voltage server
hours_to_show: 25
title: Input voltage
logarithmic_scale: true
- type: entities
entities:
- entity: sensor.myups_input_voltage
name: Input voltage
- entity: sensor.myups_battery_charge
name: Battery_charge
- entity: sensor.myups_battery_voltage
name: Battery voltage
- entity: sensor.myups_input_power_sensitivity
name: Input power sensitivity
- type: grid
cards:
- type: heading
icon: mdi:flash-triangle
heading: Voltage indicators
heading_style: title
- type: custom:power-flow-card-plus
entities:
battery:
entity: sensor.myups_battery_charge
state_of_charge: sensor.myups_battery_charge
use_metadata: false
invert_state: false
color_value: false
color:
consumption:
```

```
- 0
- 255
- 64
production:
- 0
- 98
- 255
solar:
entity: sensor.river_2_solar_in_power
display_zero_state: true
secondary_info: {}
grid:
secondary_info: {}
entity: sensor.elivco_plug_power
icon: ''
power_outage:
icon_alert: ''
clickable_entities: true
display_zero_lines:
mode: show
transparency: 50
grey_color:
- 189
- 189
- 189
use_new_flow_rate_model: true
w_decimals: 0
kw_decimals: 1
min_flow_rate: 0.75
max_flow_rate: 6
max_expected_power: 2000
min_expected_power: 0.01
watt_threshold: 1000
transparency_zero_lines: 0
title: Energy distribution in the home
- type: entities
entities:
- entity: sensor.river_2_ac_in_volts
- entity: sensor.river_2_total_in_power
- entity: sensor.river_2_ac_out_volts
- entity: sensor.river_2_total_out_power
- entity: sensor.river_2_type_c_out_power
- entity: sensor.river_2_usb_out_power
title: Power Station Info
- type: grid
cards:
- type: heading
icon: mdi:battery-alert
heading: Device battery level
heading_style: title
- type: custom:power-flow-card-plus
title: Energy distribution in the home
entities:
solar: sensor.river_2_solar_in_power
battery:
entity: sensor.ecoflow_battery_level
name: EcoFlow Battery
type: state_of_charge
individual:
- entity: sensor.elivco_plug_power
name: Elivco Plug
```

```
- entity: sensor.tuya_plug_power
name: Tuya Plug
- entity: sensor.xiaomi_plug_power
name: Xiaomi Plug
- entity: sensor.skykettle_rk_m216s_e_total_energy_consumed
name: SkyKettle Energy
layout:
battery:
combined_entity: false
entities:
- entity: sensor.ecoflow_battery_level
name: EcoFlow Battery
unit: '%'
decimals: 0
show_state_of_charge: true
icon: mdi:battery-charging
color_of_icon: '#4CAF50'
color_of_circle: '#4CAF50'
display_zero_tolerance: true
display_state: true
invert_state: false
color_of_value: '#4CAF50'
- entity: sensor.myups_battery_level
name: UPS Battery
unit: '%'
decimals: 0
show_state_of_charge: true
icon: mdi:battery-charging
color_of_icon: '#2196F3'
color_of_circle: '#2196F3'
display_zero_tolerance: true
display_state: true
invert_state: false
color_of_value: '#2196F3'
advanced:
show_entity_names: true
show_icons: true
tap_action:
action: none
- type: entities
entities:
- entity: sensor.tuya_plug_current
name: Room 1 consumption relay
- entity: switch.tuya_plug
name: Status relay room 1
- entity: sensor.elivco_plug_current
name: Room 2 consumption relay
- entity: switch.elivco_plug
name: Status relay room 2
- entity: sensor.xiaomi_plug_current
name: Room 3 consumption relay
- entity: switch.xiaomi_plug
name: Status relay room 3
title: Energy consumers
show_header_toggle: false
- type: grid
cards:
- type: heading
icon: mdi:lightning-bolt
heading: Energy consumers
heading_style: title
```

```
- type: custom:battery-state-card
- type: entities
entities:
- entity: sensor.river_2_battery_level
- entity: sensor.river_2_battery_volts
- entity: sensor.river_2_battery_charging_state
- entity: sensor.river_2_charge_remaining_time
- entity: sensor.river_2_remaining_time
title: Power Station Battery Info
max_columns: 4
```

Е.2. Автоматизації розділу «Battery»

Підсистема енергоменеджменту відповідає за моніторинг стану джерел безперебійного живлення (ДБЖ та портативної зарядної станції) і керування навантаженням. Наведені нижче сценарії реалізують логіку автоматичного відключення некритичних споживачів під час знеструмлення, превентивний захист від перевантаження мережі та сповіщення про критичний рівень заряду.

Автоматизація 1. Автоматичне скидання навантаження (Load Shedding).

```
alias: "Energy: Load shedding during outage"
description: "Turns off non-essential smart plugs when grid power is lost to extend EcoFlow battery runtime."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.river_2_ac_in_volts
below: 10
action:
- service: switch.turn_off
target:
entity_id:
- switch.tuya_plug
- switch.elivco_plug
- switch.xiaomi_plug
- service: notify.notify
data:
title: "Energy Management: Load Shedding"
message: "Grid power lost. Non-essential high-power relays have been disabled to conserve EcoFlow battery."
```

Автоматизація 2. Аварійне попередження про низький рівень заряду станції.

```
alias: "Energy: EcoFlow low battery alert"
description: "Notifies the administrator when the portable power station battery drops below 20 percent."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.river_2_battery_level
below: 20
action:
- service: notify.notify
data:
title: "Power Alert: EcoFlow Low Battery"
message: "The EcoFlow River 2 battery is below 20 percent. Please reduce power consumption immediately."
```

Автоматизація 3. Захист розумних розеток від перевантаження.

```
alias: "Energy: Smart plug overload protection"
description: "Automatically disables a smart plug if its power consumption exceeds safe hardware limits."
mode: parallel
trigger:
- platform: numeric_state
entity_id: sensor.tuya_plug_power
above: 2500
id: "tuya_overload"
- platform: numeric_state
entity_id: sensor.elivco_plug_power
above: 2500
id: "elivco_overload"
action:
- choose:
- conditions:
- condition: trigger
id: "tuya_overload"
sequence:
- service: switch.turn_off
target:
entity_id: switch.tuya_plug
- service: notify.notify
data:
title: "Safety Alert: Plug Overload"
message: "Room 1 relay exceeded 2500W and was shut off to prevent thermal damage."
- conditions:
- condition: trigger
id: "elivco_overload"
sequence:
- service: switch.turn_off
target:
entity_id: switch.elivco_plug
- service: notify.notify
data:
title: "Safety Alert: Plug Overload"
message: "Room 2 relay exceeded 2500W and was shut off to prevent thermal damage."
```

Автоматизація 4. Попередження про критичний залишок часу автономної роботи.

```
alias: "Energy: Critical remaining runtime"
description: "Warns the user to gracefully shut down connected IT equipment if battery time drops below 15 minutes."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.river_2_remaining_time
below: 15
action:
- service: notify.notify
data:
title: "Critical Alert: Imminent Shutdown"
message: "Less than 15 minutes of battery runtime remaining. Initiate graceful shutdown of all connected devices."
```

Автоматизація 5. Сповіщення про повне відновлення електроенергії.

```
alias: "Energy: Power station fully charged"
description: "Sends a notification when the EcoFlow reaches 100 percent charge capacity after grid restoration."
mode: single
trigger:
- platform: numeric_state
```

```
entity_id: sensor.river_2_battery_level
above: 99
action:
- service: notify.notify
data:
title: "Energy Status: Fully Charged"
message: "EcoFlow River 2 is fully charged and ready for backup operations."
```

Автоматизація 6. Багаторівневе скидання навантаження (Tiered Load Shedding).

```
alias: "Energy: Tiered load shedding at 50 percent"
description: "Disables secondary priority plugs when the portable station drops below 50 percent during an outage,
   reserving power for critical IT infrastructure."
mode: single
trigger:
- platform: numeric_state
entity_id: sensor.river_2_battery_level
below: 50
condition:
- condition: numeric_state
entity_id: sensor.river_2_ac_in_volts
below: 10
action:
- service: switch.turn_off
target:
entity_id:
- switch.xiaomi_plug
- service: notify.notify
data:
title: "Energy Management: Tier 2 Shedding"
message: "Battery dropped below 50 percent. Secondary devices (Room 3) disabled to extend server runtime."
```

Автоматизація 7. Щоденний аналітичний звіт з енергоспоживання.

```
alias: "Energy: Daily consumption report"
description: "Generates a daily energy report comparing today's usage with yesterday's and dynamically lists the top 3
    consumers using Jinja2 templates."
mode: single
trigger:
- platform: time
at: "23:55:00"
action:
- service: notify.notify
data:
title: "Daily Energy Analytics"
message: >
{% set total_today = states('sensor.daily_total_energy') | float(0) %}
{% set total_yesterday = state_attr('sensor.daily_total_energy', 'last_period') | float(0) %}
{% set diff = (total_today - total_yesterday) | round(2) %}
{% set diff_text = "more" if diff > 0 else "less" %}

Today's total energy consumption is {{ total_today }} kWh.
This is {{ diff | abs }} kWh {{ diff_text }} than yesterday ({{ total_yesterday }} kWh).

Top 3 consumers today:
{% set consumers = [
  ('Room 1 Plug', states('sensor.daily_tuya_plug_energy') | float(0)),
  ('Room 2 Plug', states('sensor.daily_elivco_plug_energy') | float(0)),
  ('Room 3 Plug', states('sensor.daily_xiaomi_plug_energy') | float(0)),
  ('Smart Kettle', states('sensor.daily_kettle_energy') | float(0))
```

```
  ] %}
{% set sorted_consumers = consumers | sort(attribute='1', reverse=true) %}

1. {{ sorted_consumers[0][0] }}: {{ sorted_consumers[0][1] }} kWh
2. {{ sorted_consumers[1][0] }}: {{ sorted_consumers[1][1] }} kWh
3. {{ sorted_consumers[2][0] }}: {{ sorted_consumers[2][1] }} kWh
```