Центральноукраїнський національний технічний університет
Факультет будівництва, транспорту та енергетики
Кафедра «Автоматизації виробничих процесів»

«Допущено до захисту»
Зав. кафедри АВП
к.т.н., доцент
_____Олександр ДІДИК
“___“ _____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти

на тему:

Дослідження та програмна реалізація системи хмарного керування пристроями комплексу рішень «розумний дім»

Виконав здобувач ІІ курсу групи АК-24М
ОПП «Автоматизація та комп’ютерно-інтегровані технології»
спеціальності 174 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка»
_____ Вадим Вдовенко
«___»_____ 2025 р.

Керівник проекту
доцент, канд.техн.наук
_____ Віктор БОСЬКО
«___»_____ 2025 р.

Рецензент
_____ Іван САВЕЛЕНКО
«___»_____ 2025 р.

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *будівництва, транспорту та енергетики*
Кафедра *автоматизації виробничих процесів*
Рівень вищої освіти *магістр*
Галузь знань *17 Електроніка, автоматизація та електронні комунікації*
Спеціальність *174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»*
Освітньо-професійна програма *«Автоматизація та комп'ютерно-інтегровані технології»*

ЗАТВЕРДЖУЮ

Завідувач кафедри Дідик О.К.

_____

"___" ___2025 року

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Вдовенка Вадима Сергійовича
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи хмарного керування пристроями комплексу рішень «розумний дім»*
2. Керівник роботи *Босько Віктор Васильович, канд. техн. наук, доцент,*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

3. Строк подання студентом роботи до захисту *02.12.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи *Метою розробки є програмне дослідження та програмна реалізація систем хмарного керування пристороями комплексу рішень «розумний дім» 1. Призначення системи; 2. Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми магістерської роботи; 3. Опис функціонування системи; 4. Реалізація роботи. розрахунки і експериментальні дані, що підтверджують вірність проектних та програмних рішень. впровадження системи в промислову експлуатацію.*

5. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Жесан Р.В.*		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Сучасний стан технології і засобів автоматизації для систем хмарного керування розумними будинками*	*15.10.25*	
2.	*Аналіз об'єкту управління та розробка структурної схеми хмарної системи керування*	*30.10.25*	
3.	*Перегляд аналогічних існуючих систем*	*05.11.25*	
3.	*Опис і обґрунтування проектних рішень*	*15.11.25*	
4.	*Практична реалізація системи хмарного управління розумним будинком*	*30.11.25*	

Дата видачі завдання 01.09.2025 р.

Керівник роботи _____ І.А. Березюк «____» _____2025 р.

Завдання прийнято до виконання

Здобувач _____ О.О. Пальонний «____» _____2025 р.

АНОТАЦІЯ

на випускну кваліфікаційну роботу студента групи АК-24М Вдовенка Вадима Сергійовича зі спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» на тему: «Дослідження та програмна реалізація системи хмарного керування пристроями комплексу рішень «розумний дім»».

Випускна кваліфікаційна робота присвячена дослідженню та програмній реалізації системи хмарного керування пристроями комплексу рішень «Розумний дім».

У роботі розглянуто сучасні технології автоматизації житлових приміщень, методи побудови систем дистанційного моніторингу та управління електронними пристроями через хмарні сервіси. Проведено аналіз існуючих систем-аналогів, визначено їх переваги та недоліки. Запропоновано власну архітектуру системи керування на основі мікроконтролерів IoT-класу з підтримкою бездротових протоколів Wi-Fi та MQTT.

Розроблене програмне забезпечення забезпечує взаємодію користувача з пристроями «розумного дому» через веб-інтерфейс і мобільний додаток, підтримує функції автоматизації сценаріїв, моніторингу стану датчиків, сповіщення про аварійні ситуації, а також передачу даних до хмарного сервера. Особливу увагу приділено питанням безпеки передавання даних і надійності роботи системи.

Об'єктом дослідження є процес забезпечення управлінням «розумним будинком» за допомогою хмарних технологій.

Предметом дослідження є методи забезпечення управління системами розумного будинку.

Отримані результати можуть бути використані для створення масштабованих систем управління об'єктами житлової та виробничої

інфраструктури, а також у навчальному процесі при підготовці фахівців з інформаційних технологій, електроніки та автоматизації.

Робота має практичне значення, оскільки запропоновані рішення можуть бути використані при розробці побутових IoT-систем та навчанні студентів спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка».

Ключові слова: розумний будинок, Інтернет речей, автоматизація, система управління, мікроконтролер, хмарні технології, дистанційне керування.

ABSTRACT

On final final qualification work of the student of the AK-24M group Vdovenko Vadim Serhiyovych in the specialty 174 "Automation, computer-integrated technologies and robotics" on the topic: "Research and software implementation of the cloud control system for devices of the "Smart Home" solution complex".

The final qualification work is dedicated to the research and software implementation of the cloud control system for devices of the "Smart Home" solution complex.

The work considers modern technologies for automation of residential premises, methods for building remote monitoring systems and controlling electronic devices via cloud services. An analysis of existing analogue systems was conducted, their advantages and disadvantages were identified. The own architecture of the control system based on IoT-class microcontrollers with support for Wi-Fi and MQTT wireless protocols was proposed.

The developed software provides user interaction with smart home devices via a web interface and mobile application, supports the functions of scenario automation, sensor status monitoring, emergency notification, and data transfer to a cloud server. Particular attention is paid to the issues of data transfer security and system reliability.

The object of the study is the process of ensuring the management of a smart home using cloud technologies.

The subject of the study is methods for ensuring the management of smart home systems.

The results obtained can be used to create scalable management systems for residential and industrial infrastructure facilities, as well as in the educational process when training specialists in information technology, electronics, and automation.

The work is of practical importance, since the proposed solutions can be used in the development of household IoT systems and in the training of students in specialty 174 "Automation, computer-integrated technologies and robotics".

Keywords: smart home, Internet of Things, automation, control system, microcontroller, cloud technologies, remote control.

ЗМІСТ

ВСТУП

Актуальність теми. Протягом останніх десятиліть технології дедалі глибше інтегруються у повсякденне життя людини. Від інтернет-банкінгу до онлайн-торгівлі - цифрові сервіси зробили побут значно зручнішим і ефективнішим. Сьогодні аналогічна технологічна революція охопила й житлову сферу, що дістала назву «розумний дім» (Smart Home).

Під поняттям «розумний дім» розуміють систему, у якій практично всі елементи житла - від освітлення, опалення та водопостачання до охоронних систем і навіть штор - можуть керуватися автоматизовано або дистанційно. Використання інтелектуальних технологій забезпечує зручність, енергоефективність і безпеку, а також дає можливість власнику контролювати стан помешкання з будь-якої точки світу.

Системи «розумного дому» спрощують повсякденні завдання, усувають необхідність великої кількості проводів і пультів, оптимізують роботу побутової техніки та підвищують рівень безпеки житла. Зокрема, інтегровані системи відеоспостереження, датчики диму, затоплення чи руху дозволяють оперативно реагувати на небезпечні ситуації. Таким чином, концепція Smart Home не лише підвищує комфорт, а й формує новий рівень керованості домашнім середовищем.

Впровадження технології «розумного дому» може відбуватися поступово - від окремих елементів, як-от голосове керування освітленням або шторами, до створення єдиного інтегрованого центру керування усіма підсистемами.

Ідея Smart Home почала формуватись у США ще у 1950-х роках, коли з’явилися перші спроби автоматизації побутових процесів. У 1970-х роках Вашингтонський інститут інтелектуального будинку сформулював сучасне визначення поняття, яке передбачає здатність системи класифікувати ситуації, адаптуватися до них і самостійно приймати рішення. 1978 рік став ключовим

етапом у розвитку концепції - саме тоді шотландська компанія Pico Electronics розробила стандарт X10, що дозволив керувати побутовими приладами через електромережу. Цей прорив започаткував активний розвиток smart-технологій, зокрема альтернативних протоколів EIB (KNX), LonTalk, IEC61158 та інших.

За останні десятиліття інтелектуальні системи управління досягли значного прогресу: від автоматичного контролю мікроклімату, енергоспоживання й безпеки до появи численних smart-ґаджетів, що формують комплексну інноваційну екосистему сучасного житла.

Мета й завдання дослідження. Метою даної роботи є дослідження та програмна реалізація системи хмарного керування пристроями комплексу «Розумний дім».

Для досягнення поставленої мети визначено такі основні завдання:

– здійснити аналіз існуючих систем хмарного керування для комплексів типу «розумний дім»;

– дослідити принципи побудови та функціонування систем хмарного керування інтелектуальними будинками;

– розробити програмне забезпечення, яке реалізує систему хмарного керування розумним будинком.

Об’єктом дослідження є процес хмарного керування інтелектуальними системами типу «розумний дім».

Предметом дослідження є методи, алгоритми та засоби реалізації систем хмарного керування розумним будинком.

Методи дослідження. У процесі роботи використано методи оптимізації, математичної статистики, а також методи системного аналізу, математичного моделювання, теорії автоматичного регулювання.

Наукова новизна

У ході виконання дослідження отримано такі результати:

– удосконалено систему хмарного керування розумним будинком;

– проведено огляд і аналіз сучасних технологій хмарного керування у системах типу Smart Home;

– розроблено вітчизняний програмний продукт, який забезпечує розширені функціональні можливості порівняно з наявними аналогами.

Практична значущість результатів полягає у тому, що запропоновані алгоритми та методи дозволяють ефективно реалізовувати системи хмарного керування технологіями «розумний будинок» з підвищеною гнучкістю, безпекою та масштабованістю.

Достовірність отриманих результатів підтверджується теоретичними викладеннями, результатами комп’ютерного моделювання, дослідженням параметрів у функціонуючій обчислювальній мережі, а також співставленням отриманих результатів із даними, наведеними в науковій літературі.

Отже, дослідження та програмна реалізація системи хмарного керування розумним будинком є актуальним і практично значущим завданням, спрямованим на розвиток сучасних технологій інтелектуальної автоматизації житлового середовища.

Основні результати досліджень викладені в одній науковій публікації.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1. Призначення системи

Smart Home (домашня автоматизація, розумний будинок) є одним із найперспективніших напрямів розвитку сучасних інформаційних та комунікаційних технологій. Завдяки таким системам усі електроприлади будівлі об’єднуються у єдину функціональну мережу, якою можна керувати централізовано - як безпосередньо користувачем через дисплей або мобільний пристрій, так і автоматично відповідно до заданих алгоритмів.

Метою роботи є побудова системи хмарного керування розумного будинку, яка складається з двох основних компонентів:

– пристрою моніторингу та контролю (Smart Switch) - модуля, що здійснює збір і передачу даних про енергоспоживання або генерацію конкретного приладу домогосподарства та забезпечує можливість його комутації;

– сервера (Smart Dispatcher) — центрального елемента системи, який акумулює дані від усіх пристроїв локальної мережі та приймає рішення щодо зміни конфігурації системи, підключення чи відключення окремих приладів залежно від поточного стану мережі.

Запропонована архітектура дозволяє забезпечити ефективне використання електроенергії, підвищити рівень автономності системи та створити основу для подальшого впровадження інтелектуальних алгоритмів енергоменеджменту.

Очікуваний результат

Результатом виконання роботи є ефективна, надійна та безпечна система управління розумним будинком з застосуванням хмарних технологій, що відповідає сучасним вимогам технологічного розвитку та енергоефективності.

1.2 Огляд засобів автоматизації та область застосування

Характеристика розроблюваної системи

Розроблена система хмарного керування розумним будинком реалізує основні концепції сучасних технологій Smart Home та забезпечує інтеграцію різних побутових пристроїв у єдину керовану екосистему.

Особливістю запропонованої системи є наявність специфічного функціоналу, який відрізняє її від існуючих аналогів, а саме:

- простота архітектури та мінімалістичний інтерфейс, що забезпечує високу надійність і доступність;
- низька вартість розробки та впровадження, що робить систему придатною для широкого кола користувачів;
- зручність розгортання — система не потребує спеціальних знань чи складного налаштування;
- автономний режим роботи за замовчуванням, із можливістю підключення до хмарного сервісу для розширеного функціоналу.

Таким чином, запропонована система поєднує простоту використання, надійність і масштабованість, що робить її ефективним рішенням для автоматизації житлових об’єктів різного типу

Огляд існуючих систем

Одним із найпомітніших сучасних трендів у сфері житлових технологій є система «Розумний будинок» (Smart Home), яка набуває все більшої популярності у світі. Ідея автоматизованого житла, що здатне самостійно контролювати побутові процеси - від приготування їжі до забезпечення безпеки - поступово стає реальністю.

У країнах Західної Європи такі системи вже впроваджено приблизно у 10% домогосподарств, тоді як в Україні цей напрям лише починає активно розвиватися. Основною причиною повільного поширення є висока вартість

обладнання та впровадження, проте тенденція до здешевлення технологій і підвищення енергоефективності стимулює інтерес з боку власників житла.

Особливу увагу системи Smart Home викликають у власників приватних будинків і великих квартир, для яких питання економії на опаленні, електроенергії та водопостачанні має суттєве значення. Енергоефективність, комфорт і безпека - це три ключові чинники, що визначають попит на подібні рішення.

Загальна характеристика системи

Система «Розумний будинок» - це комплекс датчиків і пристроїв автоматизації, які забезпечують контроль і регулювання життєдіяльності приміщення. До складу системи входять:

- кліматичні датчики, що реагують на зміну температури та вологості;
- світлові датчики, які враховують час доби;
- датчики руху та присутності, що фіксують сторонню активність у приміщенні;
- виконавчі пристрої, які регулюють роботу опалення, освітлення, водонагріву тощо.

Усі елементи системи інтегруються в єдиний центр керування, який передає інформацію на планшет або настінний монітор у будинку та синхронізується зі смартфонами користувачів. Це дозволяє дистанційно контролювати й змінювати налаштування системи. Наприклад, власник може активувати опалення за годину до повернення додому, щоб забезпечити комфортну температуру в момент прибуття.

Таким чином, технологія «Розумного будинку» поєднує автоматизацію, енергоефективність та інформаційні технології, створюючи безпечне й комфортне житлове середовище. Незважаючи на високу початкову вартість, її впровадження є перспективним напрямом розвитку сучасних житлових систем, особливо в умовах зростання цін на енергоресурси

Особливості проєктування та встановлення системи «Розумний дім»

Монтаж системи «Розумний дім» доцільно виконувати на етапі капітального або комплексного ремонту, що дозволяє приховати всі комунікації та забезпечити естетичний вигляд приміщення. Проєктування та встановлення подібних систем повинні виконувати кваліфіковані технічні фахівці, які мають відповідну підготовку та сертифікати, що підтверджують їхню компетентність у галузі автоматизації житла.

Встановлення системи силами звичайного електрика, навіть високої кваліфікації, не рекомендується, оскільки реалізація інтелектуальних мереж керування потребує спеціальних знань з програмування, схемотехніки та мережевих технологій. Так само самостійне встановлення системи за принципом “у коробці” може призвести до некоректної роботи або пошкодження побутової техніки.

Компанія-інсталятор повинна не лише здійснювати монтаж, а й забезпечувати гарантійне та післягарантійне обслуговування системи. Для цього її спеціалісти проходять професійне навчання, включаючи роботу з фірмовим обладнанням і програмним забезпеченням.

Проєкт кожного «Розумного будинку» розробляється індивідуально, відповідно до життєвих сценаріїв власників. Компанія-виконавець формує технічне завдання, проектує схему розміщення сенсорів, виконавчих пристроїв та комунікацій, а також здійснює монтаж і налаштування системи.

З огляду на високу вартість комплексних рішень, вибір інсталяційної компанії є ключовим етапом. Рекомендовано звертати увагу на досвід роботи не менше п’яти років у сфері автоматизації будівель. Найкраще користуватися рекомендаціями попередніх клієнтів, адже позитивні відгуки є показником надійності виконавця.

Перевагу варто надавати системам від відомих європейських або американських виробників, що працюють на ринку понад 20 років. Наявність сертифіката KNX є важливим критерієм якості, оскільки його отримують лише перевірені виробники, які відповідають міжнародним стандартам у галузі інтелектуального керування будівлями.

Інтернет речей та хмарні обчислення

Хмарні технології та Інтернет речей у системах «Розумний дім»

На сучасному етапі розвитку технологій більшість систем «розумного будинку» не забезпечують повноцінного віддаленого управління через мережу Інтернет. Водночас мобільні пристрої з постійним доступом до мережі стали звичним атрибутом повсякденного життя, що створює передумови для розвитку віддалених систем керування побутовими пристроями.

Концепція Інтернету речей

У 1999 році засновник дослідницького центру Auto-ID Center Массачусетського технологічного інституту Кевін Ештон запропонував термін Internet of Things (Інтернет речей). Його суть полягає у створенні середовища, де фізичні об’єкти, оснащені сенсорами та бездротовими модулями (Wi-Fi, Bluetooth тощо), здатні взаємодіяти між собою та з користувачами через глобальну мережу.

Розвиток концепції IoT забезпечив технічну можливість віддаленого управління системами “Розумний дім”, де смартфони, планшети, телевізори, датчики та контролери можуть обмінюватися даними в реальному часі.

Переваги віддаленого управління

Головною перевагою інтеграції функцій віддаленого керування є підвищення рівня безпеки житла. Користувач може спостерігати за обстановкою в домі за допомогою камер або контролювати стан датчиків (пожежних, руху, відкриття дверей тощо), перебуваючи поза межами помешкання.

Крім того, дистанційний доступ є зручним для користувачів, які забувають вимкнути побутові прилади або освітлення.

Ще однією важливою перевагою є зростання комфорту. Користувач може заздалегідь увімкнути опалення, кондиціонер чи освітлення перед поверненням додому. Таким чином, функція віддаленого доступу підвищує енергоефективність та індивідуалізує управління побутом.

Використання хмарних обчислень

Реалізація віддаленого доступу стає можливою завдяки хмарним обчисленням, які забезпечують користувачеві доступ до мережевих ресурсів, сервісів і застосунків через Інтернет.

Найбільш відповідною моделлю для систем «розумного будинку» є SaaS (Software as a Service) - «програмне забезпечення як послуга». Вона дозволяє користувачеві працювати з додатком без потреби в його локальному встановленні, а всі оновлення та обслуговування виконуються на стороні хмарного сервера.

У практиці впровадження Smart Home можливі два основні підходи:

Хмарний контролер. Керування пристроями здійснюється через хмарну платформу, що дозволяє взаємодіяти із системою з будь-якої точки світу при наявності доступу до Інтернету.

Локальний контролер із хмарною підтримкою. Основне керування відбувається локально, а хмара використовується лише для віддаленого доступу та обміну даними. У цьому випадку знижується навантаження на апаратні ресурси контролера, оскільки більшість обчислень виконується на сервері.

Переваги гібридного підходу

Другий варіант є більш універсальним, оскільки не потребує повної заміни вже встановленого обладнання. Для інтеграції хмарного доступу достатньо забезпечити контролеру підключення до Інтернету та конфігурацію зв’язку з хмарним сервером. Такий підхід дозволяє модернізувати існуючі системи Smart Home із мінімальними фінансовими та технічними затратами.

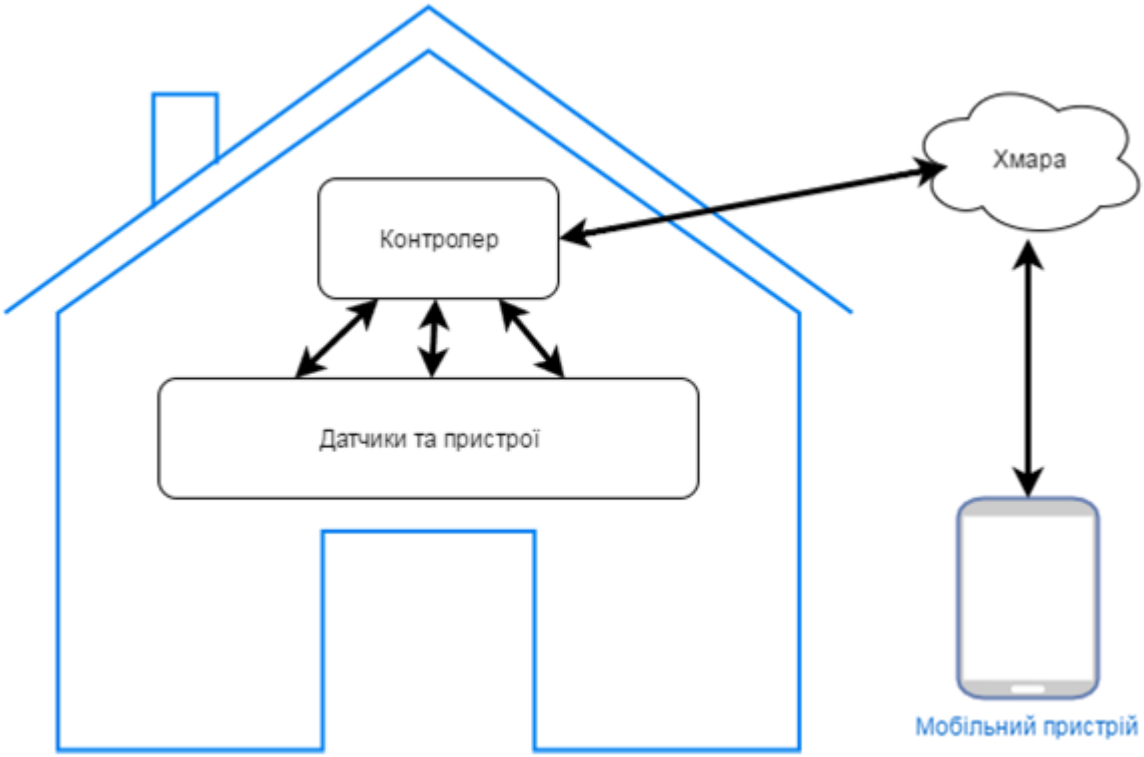

Рисунок 1.1 – Контролер розташовуваний в будинку

Протоколи взаємодії у хмарних системах «Розумний дім»

Віддалене керування системами «розумного будинку» може здійснюватися через веб-браузер або за допомогою спеціального мобільного застосунку, який забезпечує інтуїтивний доступ до функцій управління пристроями в реальному часі.

Сучасні компоненти систем Smart Home - як кінцеві пристрої (датчики, побутові прилади), так і контролери - часто використовують власні протоколи обміну даними та можуть взаємодіяти з хмарними сервісами лише через інтерфейси прикладного програмування (API). Це створює проблему сумісності: не завжди можливо інтегрувати до вже наявної системи пристрої інших виробників, що працюють за різними протоколами.

Використання єдиного хмарного сервісу, який виступає посередником між різними пристроями, дозволяє стандартизувати взаємодію та створити уніфікований інтерфейс управління. У цьому випадку всі пристрої комунікують не безпосередньо, а через хмару, що забезпечує:

- можливість інтеграції обладнання різних виробників;
- гнучкість системи у процесі розширення;
- зниження витрат на технічне обслуговування та модернізацію.

Для ефективної взаємодії між хмарним сервером і пристроями системи необхідно забезпечити єдину мову обміну даними. Одним із найпоширеніших рішень є обмін XML-повідомленнями з використанням протоколу SOAP (Simple Object Access Protocol).

SOAP - це простий протокол доступу до об’єктів, який базується на форматі XML і дозволяє забезпечити стандартизовану, безперервну взаємодію між веб-сервісами та пристроями, навіть якщо вони працюють за різними протоколами.

Основні переваги використання SOAP у хмарних системах керування:

– можливість структурованого кодування даних у форматі XML як для простих, так і для складних об’єктів;

– наявність додаткових інструментів для реалізації функцій безпеки, трасування та розширення протоколу;

– широка підтримка серед різних мов програмування — існують готові бібліотеки SOAP для більшості сучасних платформ.

Таким чином, застосування протоколу SOAP у системах «Розумний дім» забезпечує високу сумісність, масштабованість та захищеність обміну даними між хмарними сервісами і локальними пристроями, що є ключовою вимогою до інтелектуальних систем керування.

Спрощений мережевий протокол MQTT

MQTT (Message Queue Telemetry Transport) - це спрощений мережевий протокол прикладного рівня, який працює поверх стеку TCP/IP та використовується для обміну даними між пристроями за принципом публікації та підписки (publish–subscribe).

Перша версія протоколу була розроблена у 1999 році д-ром Енді Станфорд-Кларкомом (IBM) та Арленом Ніппером (Arcom), а у 2014 році

специфікація MQTT 3.1.1 була офіційно стандартизована консорціумом OASIS.

Основні переваги MQTT

Протокол MQTT має низку технічних і функціональних переваг, що робить його одним із найзручніших рішень для систем із великою кількістю сенсорів і контролерів, зокрема у сфері Smart Home:

- використання шаблону проектування publish–subscribe, який забезпечує зручну модель обміну повідомленнями між пристроями;
- простота реалізації — це легкий програмний модуль без надлишкової функціональності, який можна інтегрувати у будь-яку складну систему;
- мінімальне навантаження на мережеві канали, що особливо важливо при передачі даних із численних IoT-пристроїв;
- стійкість до втрат зв’язку — протокол дозволяє продовжувати роботу навіть у разі тимчасових проблем на лінії зв’язку;
- гнучкість у роботі з даними — MQTT не накладає обмежень на формат передаваного контенту;
- простота адміністрування та масштабування системи;
- можливість обробки динамічних повідомлень, що не були заздалегідь визначені у системі.

Робота MQTT базується на центральному компоненті - брокері повідомлень (message broker), який відповідає за отримання повідомлень від відправників (publishers) і доставку їх підписникам (subscribers) відповідно до теми повідомлення.

Протокол визначає набір методів (так званих «дієслів»), які описують дії, що виконуються над певними ресурсами. Ці ресурси можуть бути як статичними (наприклад, файли або дані з бази), так і динамічними (генеруються сервером у процесі обробки).

Однією з важливих характеристик MQTT є прапор Retain, який дозволяє зберігати останнє опубліковане повідомлення на сервері. Завдяки

цьому новий клієнт одразу отримує актуальні дані при підключенні, навіть якщо не був у мережі під час їх публікації. Це забезпечує надійність передачі даних у випадку нестабільного з’єднання.

Аналіз еволюції протоколів обміну даними показує, що MQTT став одним із ключових стандартів для реалізації інтернету речей (IoT) і систем розумного будинку. Його простота, універсальність і стійкість до мережевих збоїв роблять його оптимальним рішенням для інтеграції сенсорних мереж, контролерів і хмарних сервісів.

Разом з тим, розвиток IoT-простору триває. Великі постачальники хмарних платформ - такі як AWS, Google Cloud, Microsoft Azure - активно вдосконалюють власні MQTT-брокери, додаючи підтримку розширених функцій безпеки, аналітики та штучного інтелекту. Подальші напрями розвитку цих технологій розглядаються у наступному розділі.

Технології розумного дому і автоматизації стають все більш популярними. Гіганти хмарних обчислень все частіше починають дивитися у сторону Інтернету речей та пропонують все більшу кулькість технологій для зручного впровадження автоматизації для кожного.

AWS IoT (Amazon)

AWS IoT (Amazon Web Services Internet of Things) - це масштабована хмарна платформа, що забезпечує підключення, керування та взаємодію пристроїв Інтернету речей з іншими сервісами екосистеми AWS.

Платформа дозволяє:

- підключати IoT-пристрої до хмарної інфраструктури та між собою;
- захищати дані та забезпечувати безпечну взаємодію між пристроями, додатками та сервісами за допомогою механізмів автентифікації, шифрування й керування доступом;
- здійснювати обробку даних у режимі реального часу або за розкладом, використовуючи сервіси AWS Lambda, AWS Analytics чи Amazon Kinesis;

- реагувати на події через запуск правил та сценаріїв, визначених користувачем;
- забезпечувати безперервну роботу пристроїв і додатків навіть у разі відсутності прямого підключення до Інтернету - завдяки підтримці локальної взаємодії через AWS IoT Greengrass.

Таким чином, AWS IoT виступає централізованим середовищем для розробки, моніторингу й керування системами розумного будинку та іншими IoT-рішеннями, поєднуючи високу продуктивність, масштабованість і гнучкі інструменти безпеки.

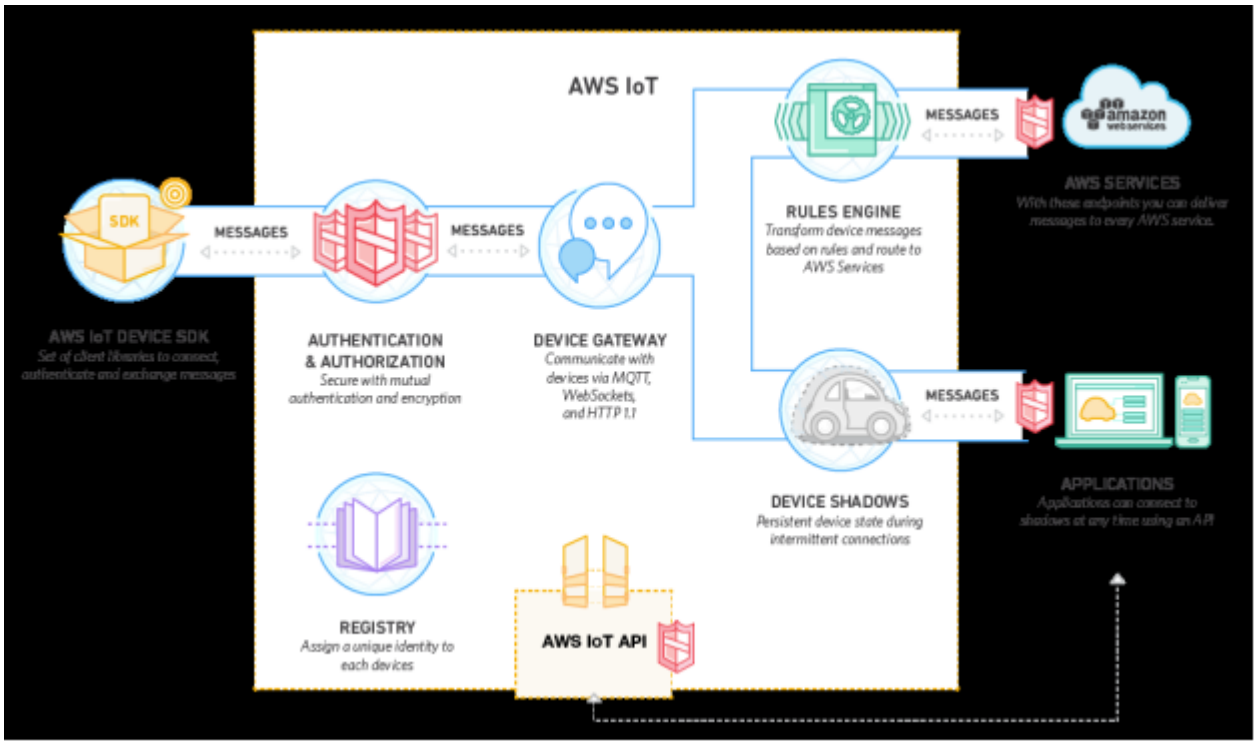

Рисунок 1.2 – Схема роботи AWS IoT.

AWS IoT SDK для пристроїв

Платформа AWS IoT надає набір інструментів розробника (Software Development Kit, SDK), що забезпечують швидке та зручне підключення апаратних пристроїв і мобільних застосунків до хмарної інфраструктури Amazon Web Services.

AWS IoT Device SDK реалізує основні функції підключення, автентифікації та обміну даними між пристроями та хмарною платформою за допомогою таких протоколів, як MQTT, HTTP та WebSockets.

SDK підтримує популярні мови програмування, зокрема C, JavaScript та Arduino, і включає:

- клієнтські бібліотеки для інтеграції з апаратними або вбудованими системами;
- керівництво розробника (Developer Guide) з детальним описом процесу налаштування та роботи з платформою;
- посібник з перенесення (Porting Guide) для виробників, що адаптують SDK до власних апаратних рішень.

Крім офіційного набору SDK, розробники можуть використовувати альтернативні бібліотеки з відкритим вихідним кодом або створювати власні програмні рішення для взаємодії з сервісами AWS IoT. Такий підхід забезпечує гнучкість і сумісність з різноманітними пристроями — від мікроконтролерів до повнофункціональних вбудованих систем.

Шлюз пристроїв

Шлюз пристроїв (AWS IoT Device Gateway) є ключовим компонентом платформи AWS IoT, який забезпечує безпечну, стабільну та масштабовану взаємодію між пристроями користувачів і хмарною інфраструктурою Amazon Web Services.

Основним завданням шлюзу є організація обміну повідомленнями між пристроями за допомогою моделі публікації–підписки (Publish–Subscribe). Така модель забезпечує як індивідуальну взаємодію (“один-до-одного”), так і групову комунікацію (“один-до-багатьох”). У межах останньої схеми підключений пристрій може транслювати дані одночасно всім підписникам певної теми (topic), що підвищує ефективність обміну інформацією у великих мережах IoT.

Шлюз пристроїв AWS IoT підтримує основні протоколи комунікації — MQTT, WebSocket та HTTP/1.1, що забезпечує сумісність із більшістю

сучасних IoT-рішень. Крім того, система дозволяє інтегрувати пропрієтарні або застарілі протоколи, що робить платформу універсальною для застосування в різних технічних середовищах.

Однією з ключових особливостей AWS IoT Device Gateway є автоматичне масштабування. Платформа здатна обробляти запити від понад мільярда пристроїв, не потребуючи додаткового розгортання чи розширення інфраструктури з боку користувача.

Таким чином, шлюз пристроїв AWS IoT виступає центральним вузлом комунікації, який гарантує безпечний, продуктивний та масштабований обмін даними між інтелектуальними пристроями, мобільними застосунками та хмарними сервісами.

Аутентифікація і авторизація

Платформа AWS IoT реалізує комплексну систему взаємної автентифікації та шифрування між усіма точками підключення, що гарантує надійний захист даних під час обміну між пристроями й хмарною інфраструктурою. Кожен канал зв'язку функціонує виключно після перевірки автентичності сторін, що виключає можливість несанкціонованого доступу.

AWS IoT підтримує кілька механізмів автентифікації, які застосовуються залежно від протоколу взаємодії:

SigV4 (Signature Version 4) — стандартний метод автентифікації AWS, що забезпечує підписування запитів для підтвердження їхньої достовірності;

сертифікати X.509 — класичний криптографічний підхід, який гарантує перевірку ідентичності пристроїв за допомогою цифрових сертифікатів.

Різні протоколи використовують відповідні методи автентифікації:

- MQTT — автентифікація виключно на основі сертифікатів X.509;
- HTTP — підтримує як SigV4, так і сертифікати X.509;
- WebSockets — автентифікація за схемою SigV4.

AWS IoT дозволяє використовувати як сертифікати, створені безпосередньо в межах сервісу, так і сертифікати, підписані стороннім центром сертифікації (CA). До кожного сертифіката може бути прив'язано

роль або політику доступу, що визначає рівень дозволів для пристрою чи застосунку. Це дає змогу відкликати або змінювати доступ без необхідності фізичного втручання в обладнання.

Користувач може створювати, оновлювати й керувати сертифікатами та політиками через AWS Management Console або за допомогою API. Такий підхід забезпечує централізоване адміністрування безпеки та спрощує масштабування системи.

Для автентифікації користувачів мобільних застосунків AWS IoT інтегрується з сервісом Amazon Cognito, який створює унікальні ідентифікатори користувачів і надає тимчасові облікові дані з обмеженими правами доступу до ресурсів AWS. Це дозволяє мобільним клієнтам безпечно взаємодіяти з платформою без збереження довгострокових ключів.

Реєстр

Реєстр пристроїв (Device Registry) є складовою частиною платформи AWS IoT, яка відповідає за ідентифікацію, облік і керування метаданими підключених пристроїв.

Реєстр забезпечує унікальне розпізнавання кожного пристрою в межах системи відповідно до стандартизованого формату ідентифікації, який не залежить від типу пристрою, його виробника чи способу підключення. Це дозволяє створювати масштабовані гетерогенні мережі IoT без втрати узгодженості даних.

Окрім ідентифікації, реєстр підтримує збереження метаданих пристроїв, що описують їхні характеристики та функціональні можливості. Наприклад, у системі можна зареєструвати температурний датчик із зазначенням, у якій шкалі (Цельсія або Фаренгейта) він працює. Такі метадані спрощують процес інтеграції різних типів пристроїв у єдину систему моніторингу та управління.

Важливо, що зберігання метаданих у реєстрі не потребує додаткових витрат. Для збереження даних у активному стані достатньо періодично звертатися до відповідного запису або оновлювати його не рідше ніж раз на

сім років. Це забезпечує довготривале зберігання інформації без необхідності втручання з боку користувача.

Таким чином, AWS IoT Device Registry є основою централізованого управління інфраструктурою “розумного дому”, забезпечуючи уніфікацію, гнучкість та довговічність даних про підключені пристрої.

Фреймворк правил

AWS IoT Rules Engine є ключовим компонентом платформи, який дозволяє створювати інтелектуальні IoT-застосунки для збору, обробки, аналізу та маршрутизації даних, що генеруються підключеними пристроями, без необхідності розгортання або адміністрування власної інфраструктури.

Движок правил оцінює вхідні повідомлення, що надходять від пристроїв у середовище AWS IoT, та виконує їх фільтрацію, трансформацію і передачу іншим пристроям або хмарним сервісам відповідно до заданих бізнес-правил. Одне правило може бути застосоване як до даних одного пристрою, так і до потоків даних від багатьох джерел, виконуючи одну або декілька дій одночасно.

Rules Engine забезпечує маршрутизацію повідомлень до різних сервісів екосистеми AWS, зокрема:

- AWS Lambda - для запуску функцій обробки подій у реальному часі;
- Amazon Kinesis - для потокової аналітики даних;
- Amazon S3 - для довготривалого зберігання великих обсягів інформації;
- Amazon DynamoDB — для збереження структурованих даних;
- Amazon Machine Learning - для побудови моделей прогнозування;
- Amazon CloudWatch - для моніторингу та збору метрик;
- Amazon Elasticsearch Service (з інтеграцією з Kibana) - для візуалізації аналітичних даних.

Крім того, через AWS Lambda, Amazon Kinesis або Amazon Simple Notification Service (SNS) забезпечується доступ до зовнішніх кінцевих точок - наприклад, сторонніх систем або API-інтерфейсів.

Гнучкість створення правил

Правила можуть створюватися через AWS Management Console або визначатися вручну з використанням SQL-подібного синтаксису, що спрощує фільтрацію й обробку даних. Користувач може задавати умови дій залежно від вмісту повідомлень. Наприклад:

- якщо показник температури перевищує встановлений поріг, система автоматично передає дані до функції AWS Lambda;
- якщо температура певного сенсора на 15 % перевищує середнє значення п’яти інших пристроїв, може виконуватися попереджувальна дія або створюватися аналітичний запис у базі даних.

Обробка та розширюваність Rules Engine

Rules Engine має широкий набір вбудованих функцій перетворення даних, які забезпечують виконання типових операцій — агрегацію, фільтрацію, математичні обчислення, логічні перевірки, форматування та конверсію даних.

Для розширення можливостей системи платформа AWS надає інтеграцію з сервісом AWS Lambda, що дозволяє виконувати довільний програмний код безпосередньо у хмарі. Завдяки цьому кількість доступних операцій обробки даних є практично необмеженою.

Наприклад, користувач може реалізувати обчислення середніх або статистичних значень для великих обсягів даних, або створити власні алгоритми обробки інформації з урахуванням специфіки підключених пристроїв. Код може виконуватися на мовах Java, Node.js або Python, що забезпечує високу гнучкість і простоту інтеграції з існуючими рішеннями.

Таким чином, поєднання Rules Engine та AWS Lambda створює потужний інструментарій для побудови інтелектуальних, масштабованих і

високопродуктивних систем обробки IoT-даних, здатних адаптуватися до динамічних умов функціонування «розумного будинку» чи інших IoT-рішень.

Google Cloud Platform (GCP) пропонує комплексні рішення для побудови, моніторингу та адміністрування систем Інтернету речей. Хмарна екосистема Google забезпечує надійність, масштабованість та інтеграцію з численними сервісами для збору, зберігання, аналізу й візуалізації даних, отриманих від IoT-пристроїв.

Google Cloud Monitoring

Google Cloud Monitoring надає інструменти для спостереження за станом системи, створення панелей моніторингу та налаштування автоматичних сповіщень. Для пристроїв на базі Linux доступний Cloud Monitoring Agent (раніше Stackdriver Agent), який збирає системні метрики та передає їх у хмару. Крім того, розробники можуть використовувати Cloud Monitoring API для інтеграції власних метрик і створення кастомних панелей управління. Це дозволяє в реальному часі відстежувати навантаження, продуктивність і доступність IoT-пристроїв, своєчасно реагуючи на відхилення.

Google Cloud Logging

Google Cloud Logging забезпечує централізований збір, зберігання та аналіз журналів подій від усіх компонентів системи. Інструмент дає змогу переглядати, шукати, фільтрувати та експортувати записи журналів, що значно спрощує відлагодження й аудит роботи пристроїв. Використання Cloud Logging дозволяє уникнути створення власних систем логування, зменшити витрати часу на розробку та підвищити надійність зберігання даних.

Google Cloud Audit Logs

Google Cloud Audit Logs фіксує адміністративні дії та події доступу до даних, що відбуваються у хмарному середовищі. Система створює незмінні журнали аудиту, які можуть бути використані для перевірки безпеки, аналізу інцидентів або контролю відповідності політик доступу.Це забезпечує прозорість усіх дій користувачів і сервісів у межах Google Cloud Platform, що є

критично важливим для систем, які працюють із конфіденційними IoT-даними.

Cloud Pub/Sub

Google Cloud Pub/Sub - це хмарний сервіс обміну повідомленнями, який забезпечує асинхронну взаємодію між компонентами розподілених систем і є ключовим елементом архітектури IoT-рішень на базі Google Cloud.

Сервіс базується на моделі публікації-підписки (publish – subscribe), у межах якої створюються теми (topics) для передачі потоків або каналів даних. Різні компоненти застосунку можуть підписуватися на певні теми, отримуючи лише ті повідомлення, які їм потрібні, без необхідності розгортання окремих каналів зв’язку для кожного пристрою.

Такий підхід суттєво спрощує архітектуру системи й підвищує її масштабованість.

Cloud Pub/Sub тісно інтегрований з іншими сервісами Google Cloud Platform, зокрема з Cloud IoT Core, BigQuery, Cloud Storage, Dataflow та Cloud Functions, що дозволяє організовувати наскрізну обробку даних - від отримання телеметрії до її зберігання й аналітики.

Сервіс може виконувати роль шлюзу, буфера або адаптера швидкості між вхідними потоками даних і системами обробки, вирівнюючи навантаження у випадку різкої зміни інтенсивності передавання даних.

Багато IoT- пристроїв мають обмежені обчислювальні ресурси та невеликий обсяг пам’яті для зберігання телеметрії. Cloud Pub/Sub може використовуватися як буфер проміжних даних, що дозволяє згладжувати короткочасні пікові навантаження (“data spikes”), які виникають, коли велика кількість пристроїв одночасно реагує на події у фізичному середовищі.

Це забезпечує стабільну роботу системи моніторингу навіть у разі нерівномірного надходження даних.

Окрім стандартних API HTTPS REST, сервіс підтримує gRPC - високопродуктивний фреймворк з відкритим вихідним кодом, який

використовує бінарний формат повідомлень, що підвищує швидкість обміну даними та знижує затрати трафіку.

За результатами тестування продуктивності (див. рис. 1.3), при використанні Java-клієнта та публікації повідомлень обсягом 50 КВ через 9 каналів gRPC, Google Cloud Pub/Sub забезпечив максимальну пропускну здатність передачі даних від одного комп'ютера, демонструючи високу ефективність навіть а інтенсивних навантажень.

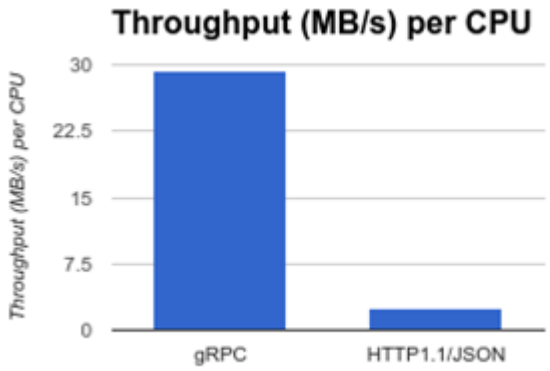

Рисунок 1.3 – Порівняльна характеристика gRPC та HTTP

Зберігання даних

Дані, що надходять із фізичного світу, можуть мати різну структуру, формат і обсяг. Хмарні платформи надають широкий спектр інструментів для їх обробки та зберігання - від неструктурованих масивів (зображення, відеопотоки, звукові дані) до структурованих таблиць, які містять показники сенсорів або результати транзакцій.

У загальному випадку стан пристрою може бути змодельований як набір пар "ключ–значення", що відображає його поточні параметри або налаштування. Деякі пристрої зберігають цей стан на рівні апаратного забезпечення, інші - у прикладному шарі.

Для систем Інтернету речей часто виникає необхідність, щоб зовнішні застосунки (наприклад, мобільний додаток чи веб-інтерфейс) мали можливість читати або змінювати актуальний стан пристрою.

Оскільки IoT-пристрої можуть перебувати у режимі енергозбереження або працювати в нестабільних мережевих умовах, важливо забезпечити синхронізацію стану з хмарою.

Завдяки такому підходу дані про стан пристрою залишаються доступними навіть у разі його тимчасової недоступності, що підвищує надійність і безперервність роботи системи. Це є основним принципом, який реалізується у багатьох сучасних IoT-платформах (зокрема, AWS IoT Device Shadow, Google IoT Core Device State тощо).

Smart Home Cloud API (Samsung)

Smart Home Cloud API - це інтерфейс прикладного програмування, який надає можливості керування та моніторингу пристроїв екосистеми Samsung Smart Home через хмарну інфраструктуру.

Завдяки компоненту Smart Home Service Control застосунок може підключатися до різних пристроїв, виконувати команди керування та надавати користувачам розширені сервіси взаємодії. Обмін даними між користувацькою хмарою та хмарою Smart Home відбувається за моделлю «cloud-to-cloud» (хмара-хмара), що забезпечує безпечну й уніфіковану інтеграцію сторонніх систем.

Samsung надає партнерам кілька REST API, які дозволяють інтегрувати власні рішення у Smart Home Cloud. Тіло запитів і відповідей REST API форматується у стандарті JSON, відповідно до Smart Home Data Model - специфікації, що описує структуру даних пристроїв та їх властивості.

Таким чином, Smart Home Cloud API забезпечує гнучку взаємодію між хмарними екосистемами, спрощує інтеграцію зовнішніх додатків і підвищує функціональність систем «розумного будинку» шляхом централізованого керування пристроями різних типів.

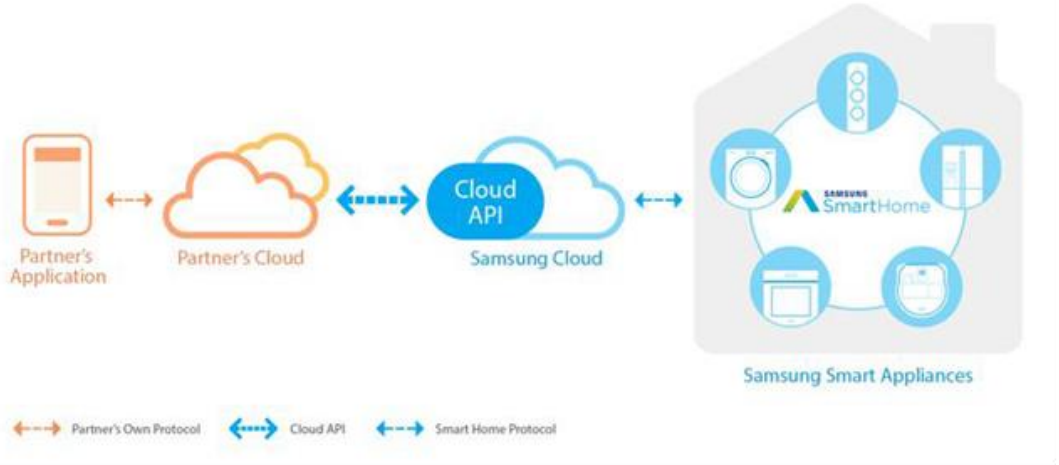

Рисунок 1.4 – Специфікації Smart Home Cloud API

Модель інтеграції

Smart Home Data Model - це структурована модель даних у форматі JSON, що використовується для представлення пристроїв екосистеми Samsung Smart Home у хмарному середовищі. Модель описує характеристики та функціональні можливості різних типів побутових пристроїв, таких як холодильники, пральні машини, кондиціонери, очищувачі повітря, робот-пилососи, сушильні машини та духові шафи. Кожен пристрій описується уніфікованим набором параметрів (тип, модель, стан, доступні ресурси), що дозволяє здійснювати централізоване керування і моніторинг у межах хмарної платформи.

Основні етапи інтеграції партнерів Smart Home Cloud:

Authentication (автентифікація). Для підключення до пристроїв Samsung партнерська система має пройти процедуру авторизації облікового запису Samsung. Взаємодія з Samsung Home Cloud можлива лише за наявності дійсного токена доступу, отриманого через протокол OAuth.

Discovery (виявлення пристроїв). Партнерська система може отримати список усіх Smart Home-пристроїв, зареєстрованих для конкретного

користувача. У результаті запиту надається детальна інформація про кожен пристрій - його тип, назву, модель, версію та доступні ресурси.

Sensing (зчитування стану). За допомогою Sensing API партнери можуть отримувати поточні дані про стан конкретного пристрою. Цей інтерфейс надає найнижчий рівень доступу до ресурсів, що дозволяє точно визначати технічний стан пристрою.

Subscription (підписка на події). Партнери можуть зареєструвати підписку на повідомлення про зміни стану пристрою, отримуючи дані у режимі реального часу при кожній зміні параметрів Smart Home-пристрою.

Notification (сповіщення). Коли Samsung Home Cloud фіксує зміни стану пристрою, вона автоматично надсилає повідомлення партнерам, що дозволяє оперативно реагувати на події та оновлення системи.

Control (керування). За допомогою Control API партнери можуть надсилати команди управління конкретним пристроям (наприклад, увімкнення кондиціонера, зміна температури, запуск пральної машини тощо).

Unsubscription (відписка). Партнер може відмовитися від отримання повідомлень для певного пристрою, після чого більше не отримуватиме сповіщення про його зміни.

Не висвітлені хмарні платформи

У межах даної роботи було проаналізовано низку хмарних платформ, призначених для керування системами «розумного будинку».

Водночас частина рішень не була розглянута через обмежену доступність технічної документації, недостатню деталізацію архітектури інтеграції або відсутність відкритих API-інтерфейсів. Зокрема, певні платформи мають закриті протоколи взаємодії, що ускладнює їхнє дослідження та використання у наукових і навчальних цілях.

Таким чином, для більш глибокого вивчення та практичної реалізації систем хмарного керування «розумним будинком» доцільно зосередитися на відкритих і добре документованих рішеннях, таких як AWS IoT, Google Cloud IoT, Microsoft Azure IoT Hub та Samsung Smart Home Cloud.

Таблиця 1.1 – Не висвітлені хмарні платформи

Назва	Посилання	Причина
EasyIoT	http://iot-playground.com/	Недостатня документація, Бета-версія
Onion.io	https://onion.io/cloud/	Незрозумілий тип інтеграції, суб'єктивна негативна оцінка
Salesforce	http://www.salesforce.com/iot-cloud/	Незрозумілий тип інтеграції, недостатня документація, суб'єктивна негативна оцінка
Oracle	https://cloud.oracle.com/iot	Незрозумілий тип інтеграції, недостатня документація, суб'єктивна негативна оцінка

1.3 Обґрунтування вибору методів розробки ситеми управління

Порівняльна характеристика хмарних платформ IoT

Для оцінки ефективності розглянутих хмарних платформ у контексті побудови систем типу «Розумний дім» проведено порівняння за низкою ключових критеріїв, що впливають на функціональність, надійність і масштабованість рішень.

Критерії оцінювання

порівняння виконано за такими основними критеріями:

- безпека передачі даних – наявність механізмів шифрування, автентифікації та контролю доступу під час обміну інформацією між пристроями та хмарною інфраструктурою;
- можливості розширення – підтримка масштабування, гнучкість архітектури, сумісність із зовнішніми сервісами та API;
- різноманітність підключених пристроїв – кількість і типи пристроїв, які можуть бути інтегровані в систему без додаткових адаптерів або шлюзів;
- моніторинг та аналіз даних – наявність інструментів для збору, обробки, візуалізації та аналітики телеметрії пристроїв.

Критерії наведено в довільному порядку; відносна вага (значущість) кожного з них буде встановлена на подальшому етапі аналізу.

Об’єкти порівняння

У дослідженні розглянуто три провідні хмарні платформи для IoT-рішень:

- Amazon IoT (AWS IoT Core);
- Google IoT (Google Cloud IoT Core);
- Samsung IoT (Samsung Smart Home Cloud).

Безпека передачі даних

Усі три платформи реалізують високий рівень захисту інформації, що передається між пристроями та хмарними сервісами. Основні методи безпеки включають:

- використання SSL/TLS-шифрування для захисту каналів зв’язку;
- автентифікацію користувачів і пристроїв через унікальні сертифікати, токени або облікові дані;
- ідентифікацію пристроїв за унікальними ідентифікаторами;
- керування політиками доступу для обмеження повноважень користувачів і сервісів.

У таблиці 1.2 подано порівняння реалізації механізмів безпеки для розглянутих платформ.

Таблиця 1.2 – Оцінка безпеки передачі даних

Назва	SSL	Аутентифікація	Унікальний ідентифікатор кожного пристрою	Додаткові засоби
Amazon IoT	+	+	+	-
Google IoT	+	+	+/-	-
Samsung IoT	+	+	+	Samsung OAuth

Google IoT (Google Cloud IoT Core) не призначає унікальний ідентифікатор для кожного окремого датчика. Натомість система формує ідентифікатор для групи пристроїв, що підключені через спільний маршрутизатор або шлюз. Такий підхід спрощує адміністрування та зменшує кількість записів у реєстрі, проте знижує рівень деталізації контролю доступу, що може бути критичним у великих системах із великою кількістю сенсорів.

Samsung IoT (Samsung Smart Home Cloud) використовує уніфіковану систему автентифікації Samsung OAuth, яка базується на єдиному обліковому записі Samsung Account для всіх пристроїв користувача. Отримання доступу до детальної технічної інформації про пристрої можливе лише через авторизований обліковий запис і за наявності самого пристрою Samsung, що забезпечує високий рівень безпеки, але обмежує можливість зовнішньої інтеграції та дослідження внутрішніх механізмів платформи.

Таким чином, Google IoT орієнтована на спрощене групове керування пристроями, тоді як Samsung IoT надає пріоритет захисту користувацьких даних і централізованому контролю в межах корпоративної екосистеми.

Можливості розширення та цінова політика хмарних платформ IoT

Розглянуті хмарні платформи відрізняються підходами до масштабування ресурсів і формування вартості послуг, що безпосередньо впливає на зручність використання, гнучкість налаштування та економічну доцільність впровадження системи Smart Home.

Amazon IoT

Платформа Amazon IoT (AWS IoT Core) надає стартовий безкоштовний пакет послуг, який включає всі основні компоненти: базу даних, обчислювальні потужності та комунікаційні канали. Основною перевагою є модель оплати за фактичне використання ресурсів (pay-as-you-go), що дозволяє планувати витрати відповідно до реальних потреб і оперативно реагувати на зміну навантаження або сезонні коливання.

AWS IoT забезпечує гнучке масштабування — користувач може незалежно збільшувати обчислювальні ресурси, обсяг пам’яті чи пропускну здатність мережі, що робить платформу придатною як для малих, так і для великих проєктів.

Google IoT

Платформа Google IoT (Google Cloud IoT Core) також пропонує стартовий безкоштовний пакет, який включає базові елементи інфраструктури: базу даних, обчислювальні потужності та канал зв’язку. Користувач може додатково підключати API-сервіси Google Cloud, які розширюють можливості системи, зокрема:

– роботу з великими даними (Big Data);

– логування та моніторинг (Cloud Logging, Cloud Monitoring);

– машинне навчання (Machine Learning API);

– мережеву оптимізацію (DNS, балансування навантаження тощо).

Хоча ці сервіси значно розширюють функціонал системи, їх вартість є досить високою, а налаштування потребує технічної кваліфікації спеціаліста. З іншого боку, їх інтеграція дозволяє скоротити час розробки та розгортання складних систем керування. Оплата додаткових послуг здійснюється з моменту їх активації, що дозволяє гнучко контролювати витрати.

Samsung IoT

Платформа Samsung IoT (Smart Home Cloud) працює за платною моделлю з моменту підключення - безкоштовна або демонстраційна версія не передбачена. Усі сервіси надаються в єдиному комплексному пакеті, який автоматично оновлюється та розширюється відповідно до виходу нових версій програмного забезпечення. Користувач не має можливості гнучкого вибору обсягу ресурсів чи налаштування тарифів, що спрощує використання, але зменшує економічну ефективність для невеликих проєктів або експериментальних систем.

Таблиця 1.3 – Оцінка можливості розширення

Назва	Безкоштовний стартовий пакет послуг	Оплата по факту використання	Можливість підключення готових додаткових API
Amazon IoT	+	+	+
Google IoT	+	-	+
Samsung IoT	-	-	-

Порівняльна характеристика хмарних платформ IoT

Різноманітність пристроїв, які можуть бути підключені

Amazon IoT (AWS IoT Core). Платформа надає SDK для Embedded C, JavaScript, а також спеціальний SDK для Arduino Yún. Оскільки більшість мікроконтролерів Arduino підтримують прошивку мовою C, це відкриває широкі можливості інтеграції системи з великою кількістю пристроїв, які можна власноруч прошити та налаштувати відповідно до потреб користувача.

Google IoT (Google Cloud IoT Core). Технологія Google Cloud Pub/Sub забезпечує підключення датчиків і мікроконтролерів через протокол зв'язку gRPC, який можна реалізувати на численних пристроях, включно з Arduino NodeMCU. Це робить платформу універсальною для побудови багаторівневих систем збору даних.

Samsung IoT (Smart Home Cloud). Платформа підтримує зв'язок лише з фірмовими пристроями Samsung, що суттєво зменшує гнучкість інтеграції та унеможливлює бюджетне тестування або використання пристроїв інших виробників.

Моніторинг та аналіз даних

Amazon IoT (AWS IoT Core). Платформа дозволяє створювати так звані «тіні» пристроїв (Device Shadows) у хмарі - це віртуальні моделі, у які записуються показники сенсорів та команди керування. Система підтримує генерацію приватних і публічних ключів для захищеної передачі даних, а також створення "ролей" (тригерів), що реагують на події: критичні показники датчиків, сигнали тривоги, команди користувача. Через відкрите API та SDK інформацію можна зчитувати за допомогою MQTT, що дає змогу легко відображати дані на клієнтських пристроях (мобільних, десктопних чи веб-інтерфейсах).

Google IoT (Google Cloud IoT Core). Платформа має розвинені засоби аналітики та візуалізації даних, включно з побудовою графіків, моніторингом у часі та підтримкою Big Data. Інтегрована нейронна мережа (Machine

Learning API) дозволяє створювати системи автоматичного керування на основі штучного інтелекту.

Ця функція поки що перебуває в бета-версії і вимагає значного обсягу навчальних даних для досягнення стабільної роботи.

Samsung IoT (Smart Home Cloud). Платформа надає спеціалізовані мобільні застосунки (на базі Android), які дозволяють отримувати інформацію з хмари без розробки власного програмного забезпечення. Передбачено систему нотифікацій із можливістю налаштування пріоритетів, підписки/відписки на події та перегляду історії показників. На жаль, через обмежений доступ до документації інші засоби моніторингу дослідити не вдалося, що ускладнює повну оцінку потенціалу платформи.

У ході аналізу були розглянуті три провідні хмарні платформи для систем Smart Home:

- AWS IoT (Amazon);
- Google Cloud IoT (Google);
- Smart Home Cloud API (Samsung).
- Порівняння здійснювалося за чотирма параметрами:
- бзпека передачі даних;
- можливості розширення;
- різноманітність підтримуваних пристроїв;
- моніторинг та аналітика даних.

Висновки за критеріями:

Безпека передачі даних. Усі три платформи забезпечують високий рівень захисту, використовуючи SSL/TLS-шифрування, автентифікацію користувачів та пристроїв і сучасні механізми керування доступом.

Можливості розширення. Перевагу має Google IoT завдяки великому набору інтегрованих сервісів (Big Data, AI, Cloud Functions). Проте більшість із них є платними, а аналоги можна реалізувати в Amazon IoT, тому вибір залежить від бюджету та кваліфікації розробника.

Різноманітність пристроїв. Samsung IoT суттєво поступається конкурентам, оскільки підтримує лише пристрої власного виробництва. Натомість Amazon IoT та Google IoT забезпечують відкритість і сумісність із широким спектром мікроконтролерів та сенсорів.

Моніторинг і аналіз даних. Усі три платформи мають достатні засоби адміністрування. Google IoT має найпотужнішу аналітичну складову, тоді як Amazon IoT забезпечує більшу інтеграційну гнучкість. Samsung IoT залишається закритою системою з обмеженими можливостями тестування.

У результаті дослідження проведено порівняльний аналіз найпопулярніших хмарних платформ IoT, які застосовуються для побудови систем «розумного будинку». Кожна з платформ має сильні та слабкі сторони, проте жодна з них не забезпечує повної відкритості, гнучкості та сумісності з бюджетними апаратними рішеннями.

Враховуючи результати аналізу, у даній магістерській роботі прийнято рішення розробити власну систему хмарного керування “Розумний дім”, яка поєднує найкращі практики провідних платформ (AWS, Google, Samsung) та адаптована до використання доступних апаратних засобів на базі NodeMCU.

Основні компоненти системи

Пристрої - електронні елементи, керування якими необхідно автоматизувати.

Датчики - сенсори збору інформації, що виступають базовими елементами системи.

Мікроконтролери - апаратні вузли, які об’єднують датчики в групи та забезпечують зв’язок із центральним процесором.

Сервер - основний обчислювальний вузол, який формує інтерфейс взаємодії користувача з системою.

Канали передачі даних - фізичні та логічні шляхи обміну інформацією, що відповідають за швидкість і безпеку комунікацій.

Хмара - зовнішній сервіс, який виконує функції бази даних і зберігає службову статистичну інформацію.

Мобільні пристрої - термінали користувача, що забезпечують віддалене керування системою.

Мова програмування C - вибрана для розробки завдяки своїй продуктивності та підтримці вбудованих систем.

Апаратна основа - мікроконтролер NodeMCU v3 LoLin, який забезпечує Wi-Fi-підключення, простоту програмування та сумісність із більшістю IoT-сенсорів.

Розгорнута постановка завдання

Відповідно до технічного завдання на магістерську роботу, реалізації підлягає програмне забезпечення, призначене для системи управління розумним будинком із підсистемою безпечної передачі даних та захистом від кібератак. Розробка спрямована на створення надійної, масштабованої та безпечної системи, яка забезпечуватиме моніторинг, контроль і автоматизацію інженерних підсистем будівлі в умовах підвищених вимог до інформаційної безпеки.

Основні завдання магістерської роботи

Технічне завдання на виконання магістерської роботи

Відповідно до технічного завдання магістерської роботи передбачено розробку програмного забезпечення, призначеного для хмарного керування системою “Розумний дім”.

У процесі виконання дослідження необхідно реалізувати такий комплекс робіт:

а) Провести детальний аналіз існуючих систем-аналогів з метою виявлення їхніх переваг і недоліків. Отримані результати аналізу слід врахувати під час проєктування та розроблення власного програмного рішення.

б) Визначити та обґрунтувати методику побудови системи контролю технологічного обладнання в автоматизованому режимі. Розробити функціональну та структурну схеми системи керування.

в) Створити програмне забезпечення, що забезпечує виконання вимог, визначених технічним завданням. Розробити блок-схеми алгоритмів основної програми та допоміжних підпрограм.

г) Реалізувати користувацький інтерфейс, який забезпечує формування та виведення на екран повідомлень про некоректні дії користувача або нестандартні ситуації у функціонуванні технологічного обладнання.

д) Підготувати рекомендації щодо організаційних і методичних заходів, необхідних для впровадження системи у промислову експлуатацію та забезпечення її ефективного використання.

е) Виконати розрахунки економічної ефективності запропонованої системи хмарного керування.

ж) Сформулювати висновки за результатами виконаної роботи, підтвердивши досягнення поставленої мети та виконання запланованого обсягу завдань.

2. ОГЛЯД ОБ'ЄКТА УПРАВЛІННЯ. СТВОРЕННЯ СТРУКТУРНОЇ ТА ФУНКЦІОНАЛЬНОЇ СХЕМИ. ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

2.1 Опис функціонування системи

Виходячи з теми магістерської роботи, необхідно розробити програмне забезпечення системи хмарного керування “Розумний дім”, а також створити автоматизовану систему управління з інтеграцією необхідних датчиків і виконавчих пристроїв.

Устаткування системи “Розумний дім”

На початковому етапі доцільно розглянути принципи перетворення звичайного житлового приміщення - квартири, дачі чи котеджу - на інтелектуальний простір, тобто “розумний будинок”. Для цього необхідно встановити в приміщенні такі основні елементи обладнання:

датчики, що здійснюють вимірювання параметрів зовнішнього середовища (температура, освітленість, вологість, наявність руху, витік води тощо).

виконавчі пристрої, які здійснюють вплив на зовнішні об’єкти згідно з отриманими командами (наприклад, реле, електроклапани, освітлення, розетки).

контролер, який обробляє дані з датчиків відповідно до заданих алгоритмів, приймає рішення і передає команди виконавчим пристроям.

Приклад структури системи

На ілюстрації (рис. 1) наведено спрощену схему системи “Розумний дім”. У ній передбачено:

- датчики контролю протікання води, встановлені у ванній кімнаті;
- датчик температури;
- датчик освітленості, розміщені у спальні;

- “розумну” розетку на кухні;
- камеру відеоспостереження, розташовану у передпокої.

Схема розташування та взаємодії охоронних датчиків наведена на рисунку 2.1.

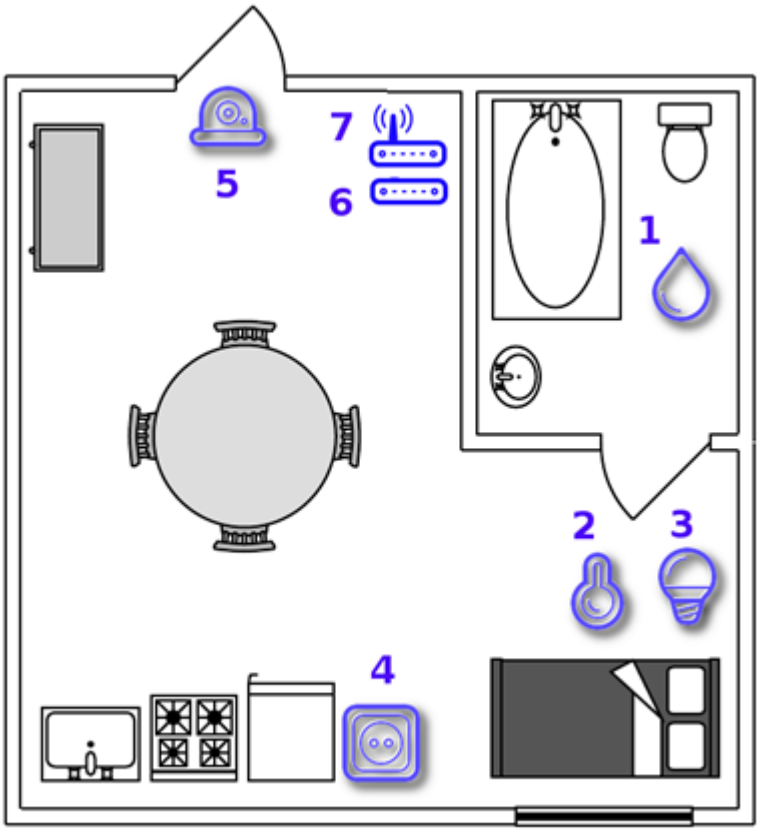

Рисунок 2.1 - Обладнання «розумного будинку»

Датчики та система зв’язку

На сучасному етапі розвитку технологій найбільшого поширення набули бездротові датчики, що працюють за комунікаційними протоколами RF433, Z-Wave, ZigBee, Bluetooth та Wi-Fi. Основними їхніми перевагами є:

зручність встановлення та експлуатації,

висока надійність роботи,

низька собівартість.

Такі пристрої активно орієнтовані на масовий ринок, що робить технологію “розумного дому” доступною для широкого кола користувачів.

Датчики та виконавчі пристрої зазвичай підключаються до контролера розумного дому (поз. 6) через бездротові інтерфейси. Контролер є спеціалізованим мікрокомп’ютером, який об’єднує всі пристрої системи в єдину мережу, координує їхню взаємодію та реалізує алгоритми керування.

Деякі сучасні рішення поєднують у собі функції датчика, виконавчого пристрою та контролера одночасно. Наприклад:

- “розумна” розетка може самостійно вмикатися або вимикатися за заданим розкладом;
- камера хмарного відеоспостереження здатна автоматично розпочинати запис відео після спрацьовування детектора руху.

У найпростіших конфігураціях можливе використання системи без окремого центрального контролера, проте для побудови гнучкої багаторівневої системи з різними сценаріями автоматизації наявність контролера є обов’язковою умовою.

Підключення до глобальної мережі. Для забезпечення віддаленого доступу та зв’язку з хмарним сервісом контролер розумного дому підключається до глобальної мережі Інтернет через стандартний роутер (поз. 7), який є звичним елементом побутової інфраструктури.

Важливою перевагою використання локального контролера є стійкість системи до втрати зв’язку з Інтернетом: навіть у разі відсутності доступу до хмарного сервісу система продовжує роботу у штатному режимі завдяки вбудованому блоку логіки, який зберігає основні сценарії керування та алгоритми взаємодії пристроїв.

Контролер розумного будинку

Контролер розумного будинку є центральним елементом системи, що забезпечує обробку даних, координацію роботи датчиків і виконавчих пристроїв, а також взаємодію з хмарним сервісом.

Конструкція контролера. Збірка контролера має просту конструкцію.

Основу складає мікрокомп’ютер (1), встановлений у пластиковий корпус (2). У корпус у відповідні роз’єми монтуються:

- карта пам’яті microSD об’ємом 8 ГБ з попередньо інстальованим програмним забезпеченням (3);
- USB-модуль мережі Z-Wave (4), що забезпечує бездротову взаємодію з периферійними пристроями.

Контролер підключається до електромережі через адаптер живлення 5 В, 2.1 А (5) за допомогою кабелю USB–microUSB (6).

Кожен пристрій має унікальний ідентифікаційний номер, який автоматично записується у конфігураційний файл під час першого запуску. Цей ідентифікатор використовується для аутентифікації контролера у хмарному середовищі та синхронізації з іншими компонентами системи.

Програмне забезпечення контролера

Програмне забезпечення розроблено автором на основі операційної системи Linux та складається з трьох основних підсистем:

Серверна підсистема - забезпечує взаємодію контролера з периферійним обладнанням “розумного дому” та хмарною інфраструктурою.

Графічний інтерфейс користувача (GUI) - дозволяє виконувати налаштування конфігурації, змінювати параметри роботи контролера, додавати або видаляти пристрої.

Підсистема зберігання даних (база даних) - призначена для фіксації станів пристроїв, збереження логіки керування та історії подій.

База даних контролера

База даних реалізована на основі вбудованої системи керування базами даних PostgreSQL і розміщується на SD-карті разом із системним програмним забезпеченням.

Основними функціями бази даних є:

- зберігання конфігурації контролера, зокрема відомостей про підключене обладнання та його поточний стан;

- підтримка логічного блоку правил (production rules), що визначають сценарії автоматизації;
- ведення індексованої інформації, зокрема посилань на локальні відеофайли, журнали подій тощо.

Після перезавантаження контролера всі дані зберігаються, що забезпечує автоматичне відновлення працездатності системи у разі збоїв живлення або непередбачених ситуацій.

Хмарний сервіс системи “Розумний дім”

Хмарний сервіс реалізує централізоване зберігання, обробку та аналітику даних, отриманих від пристроїв системи “Розумний дім”. Він забезпечує зручний, гнучкий і економічно ефективний спосіб доступу до інформації, усуваючи потребу користувача самостійно піклуватися про збереження даних.

Завдяки використанню багатопроцесорного медіасервера, оснащеного RAID-масивом із жорстких дисків ємністю 10–12 ТБ, хмарна інфраструктура значно перевищує за надійністю та обсягом пам’яті локальні носії, встановлені в контролері (microSD або Flash). Крім того, карти пам’яті мають обмежену кількість циклів перезапису і є менш надійними у довготривалій експлуатації.

Глибина (період) зберігання даних у хмарі визначається тарифним планом користувача і легко змінюється через особистий кабінет. Для доступу до хмарного сервісу не потрібно виконувати перенаправлення портів (port-forwarding) на маршрутизаторі користувача, адже пристрої “розумного дому” можуть працювати всередині приватної мережі, захищеної протоколом NAT.

Особистий кабінет доступний із будь-якого мобільного пристрою та дозволяє:

- керувати конфігурацією системи;
- задавати логіку та сценарії роботи пристроїв;
- переглядати статистику та історію подій.

Хмарна система не лише забезпечує зберігання, а й обробку інформації, що дозволяє користувачеві отримувати аналітичні звіти - наприклад, визначати

середню температуру в приміщенні за тиждень на основі даних від мультісенсорів.

2.2 Розробка структурної схеми

Структурна схема системи відображає сукупність основних елементів, з яких складається розроблювана система, а також зв’язки між ними. Її основне призначення полягає у наочному представленні складових частин системи, функціональних блоків, вузлів та каналів взаємодії між ними.

На рисунку 2.2 подано структурну схему розробленої системи, яка демонструє загальну організацію елементів “розумного дому”, включно з контролером, датчиками, виконавчими пристроями, хмарним сервером та користувацьким інтерфейсом.

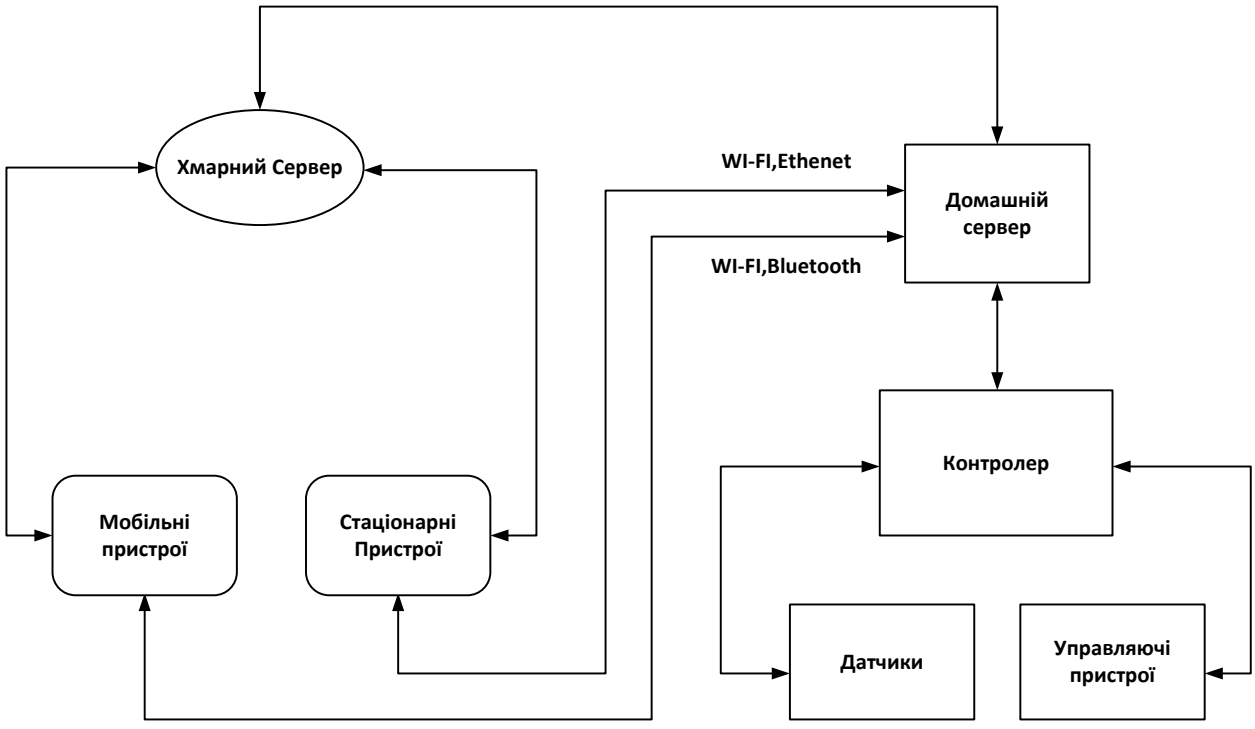

Рисунок 2.2 – Структурна схема системи «Розумний дім»

Принцип роботи пристрою

Пристрій, побудований відповідно до наведеної структурної схеми, живиться від стандартної електричної мережі змінного струму 220 В, 50 Гц. Вхідна змінна напруга за допомогою бустерного (DC-DC) конвертера перетворюється на стабілізовану постійну напругу 3.5 В, яка використовується для живлення мікроконтролера та інших низьковольтних компонентів системи.

Мікроконтролер виконує керування роботою виконавчого елемента, який здійснює комутацію електроприладу відповідно до заданих алгоритмів. У самому електричному приладі розміщені датчики напруги та струму, сигнали з яких подаються на аналогові входи АЦП мікроконтролера у вигляді аналогових напруг рівня 0 - 3.5 В. Отримані дані використовуються для контролю параметрів живлення, визначення стану навантаження та забезпечення безпечної експлуатації системи.

2.3 Розробка функціональної схеми

Функціональна схема розробленої системи наведена на рисунку 3.3.

Із рисунка видно, що система складається з таких основних функціональних блоків:

- хмара - зовнішній серверний сервіс, який виконує функції бази даних для зберігання статистичної, службової та аналітичної інформації, отриманої від пристроїв системи.
- WEB-сервер - програмний модуль, що забезпечує обробку даних, отриманих від контролера, а також надає інтерфейс взаємодії між користувачем і системою.
- контролер із навісним обладнанням - центральний елемент системи, що здійснює збір інформації з датчиків, передає її до хмари та формує сигнали керування для виконавчих механізмів.

- модуль безпеки та шифрування - підсистема, що відповідає за захист даних, автентифікацію користувача та шифрування інформаційних потоків між компонентами системи.
- Web-інтерфейс - клієнтська частина системи, що забезпечує графічне представлення даних та дає змогу користувачу керувати параметрами роботи “розумного дому” через браузер.
- мобільний застосунок - адаптований інтерфейс для смартфонів і планшетів, який забезпечує дистанційний моніторинг і керування системою в реальному часі.
- виконавчі механізми - пристрої, що безпосередньо впливають на об’єкти керування (ввімкнення/вимкнення приладів, регулювання освітлення, клімату тощо).

Веб-сервер системи

Веб-сервер є важливим програмним компонентом системи, який виконує такі функції:

- приймає та зберігає дані, отримані від Bluetooth LE-сервера;
- аналізує ці дані за допомогою алгоритмів оптимізації з метою ухвалення рішень щодо підключення або відключення побутових приладів;
- запобігає перевантаженню електромережі та потенційним аварійним ситуаціям;
- надає API-інтерфейс для ручного керування системою через графічний веб-інтерфейс користувача.

В основі веб-сервера лежить JavaScript-фреймворк Express.js, який є стандартом де-факто для більшості сучасних застосунків, створених на платформі Node.js. Express.js було обрано завдяки його високій швидкодії, простоті налаштування та ефективності обробки HTTP-запитів, що відповідає принципам фреймворку Sinatra для Ruby.

Серверний процес системи

Серверний процес є ключовим компонентом системи, що виконує основні функції з автоматизації інформаційних процесів у межах “розумного дому”. До його завдань належать:

- отримання та обробка даних від сенсорів;
- формування та передача керуючих сигналів виконавчим пристроям згідно із заданою логікою;
- взаємодія з графічним інтерфейсом і хмарним сервісом;
- виконання логічних продукційних правил, що визначають сценарії роботи системи.

Серверний процес у розробленому контролері реалізований як багатопотоковий застосунок, створений мовою C++ та запущений у системі як окремий сервіс. Такий підхід забезпечує стабільність, розширюваність і можливість паралельної обробки подій у реальному часі.

Структура серверного процесу

Основними функціональними блоками серверного процесу є:

- Диспетчер повідомлень – центральний модуль, що здійснює маршрутизацію інформаційних потоків між усіма компонентами системи.
- Сервер IP-камери – відповідає за приймання, обробку та передачу відеопотоків від камер спостереження.
- Сервер пристроїв Z-Wave – забезпечує взаємодію з бездротовими периферійними пристроями, що працюють за протоколом Z-Wave.
- Сервер продукційних логічних правил – реалізує механізми автоматизації, виконуючи правила типу «якщо... то...» для різних сценаріїв системи.
- База даних конфігурації – містить відомості про налаштування контролера, зареєстровані пристрої та набір логічних правил.
- RESTful API сервер – забезпечує взаємодію серверного процесу з графічним інтерфейсом користувача (через HTTP-запити).

- MQTT-клієнт – відповідає за обмін повідомленнями з хмарним сервером, використовуючи легкий протокол публікації/підписки.

Взаємодія між блоками

Кожен із зазначених блоків реалізовано як окремий потік, що працює у межах одного процесу. Передача інформації між потоками здійснюється у вигляді повідомлень формату JSON, або у вигляді структур даних, що відповідають цьому формату в оперативній пам’яті процесу.

Центральним елементом архітектури серверного процесу є диспетчер повідомлень, який виконує роль маршрутизатора, перенаправляючи JSON-повідомлення між усіма функціональними блоками відповідно до їхніх типів і призначення.

Типи інформаційних полів JSON-повідомлень, а також допустимі значення для кожного з них наведено в таблиці 2.3.

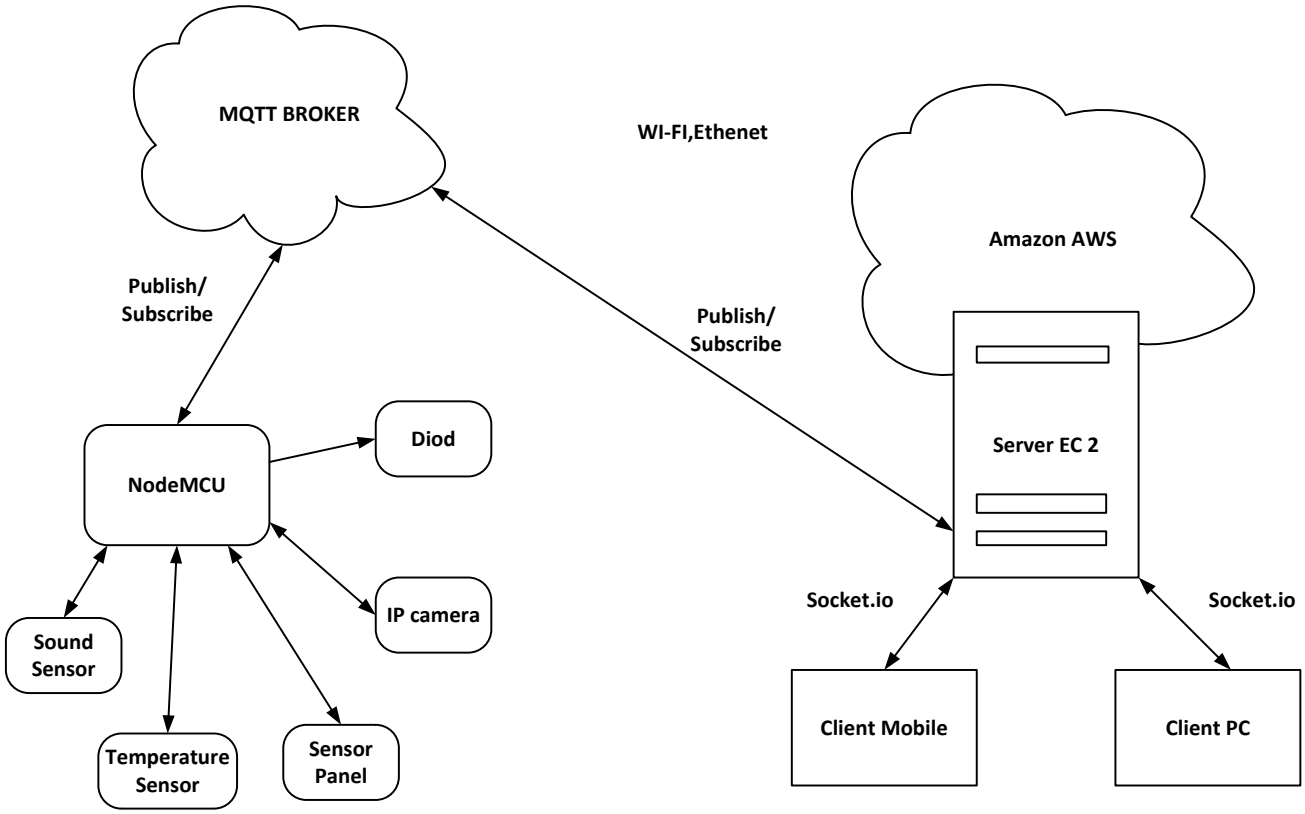

Рисунок 2.3 – Фукціональна схема системи «Розумний дім».

Розробка діаграми процесів

Діаграма процесів розробленої системи представлена на рисунку 3.4. Під час її детального аналізу можна простежити логіку функціонування системи та взаємодію між її основними складовими.

Для візуалізації процесів обробки інформації використано модель структурного проєктування, що дозволяє графічно відобразити “потоки” даних у межах інформаційної системи. Діаграма взаємодії процесів є одним із ключових інструментів структурного аналізу, оскільки дає змогу наочно представити обмін інформацією між процесами, підсистемами та зовнішніми об’єктами.

На початковому етапі проєктування розробник створює контекстну діаграму взаємодії процесів, яка показує загальний обмін даними між системою та зовнішнім середовищем. Після цього контекстна діаграма уточнюється та деталізується, утворюючи більш складну структуру з розгалуженням потоків даних і описом взаємозв’язків між окремими процесами системи.

У результаті застосування таких методів була створена система керування “розумним домом” із підсистемою забезпечення безпеки передавання керуючих сигналів від смартфона до мікроконтролера. Дана підсистема гарантує захищену передачу даних під час віддаленого управління системою за допомогою технологій Інтернету речей (IoT).

Таким чином, після розгляду опису системи, її структурної та функціональної схем, а також діаграми взаємодії процесів, доцільно перейти до опису блок-схем основної програми та підпрограм, що реалізують програмну логіку розробленої системи.

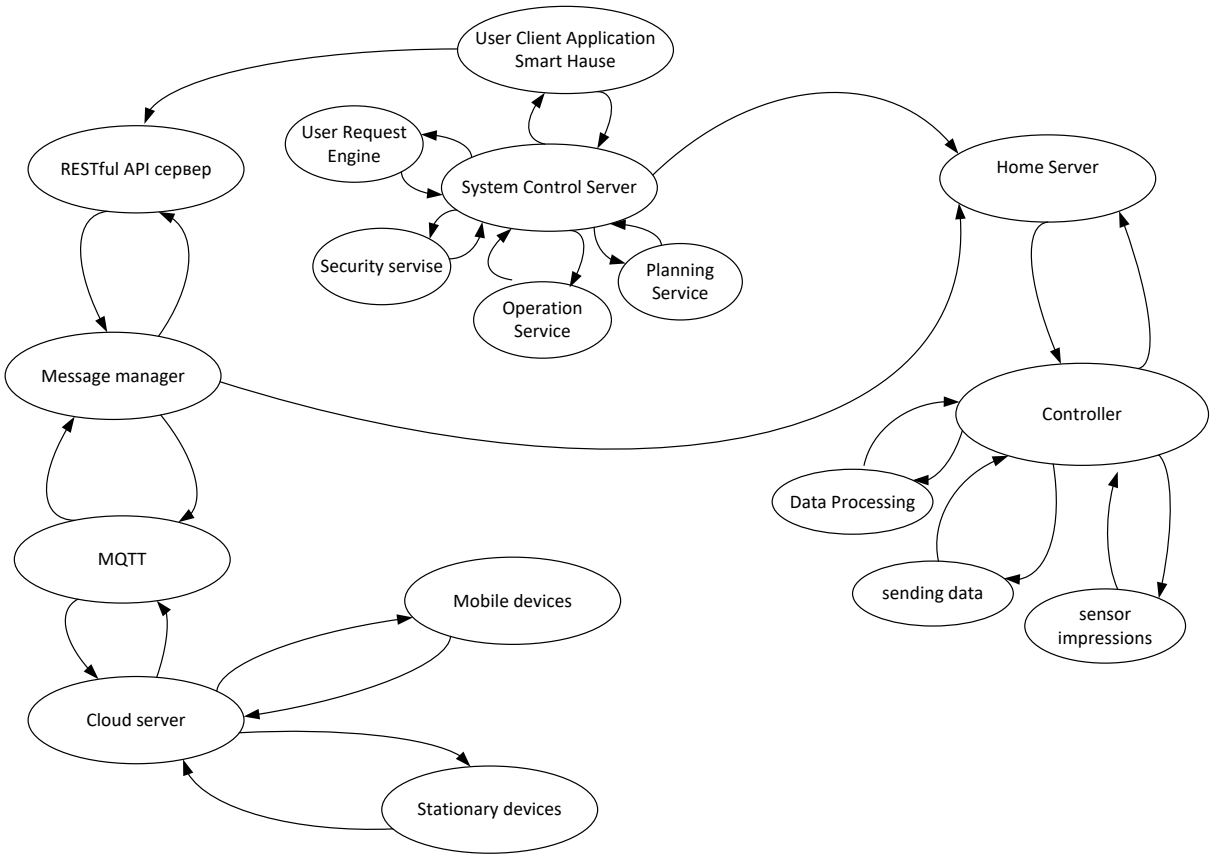

Рисунок 2.4 – Діаграма процесів системи «Розумний дім»

3 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

3.1 Реалізація системи управління розумним будинком

У ході роботи було проведено аналіз найпоширеніших і найбільш надійних хмарних платформ, призначених для реалізації систем керування “розумним домом”. Кожна з розглянутих платформ має власні переваги та обмеження, що зумовило доцільність розроблення власної хмарної системи керування, побудованої з урахуванням найкращих практик і рішень, виявлених під час аналізу існуючих сервісів.

Схема підключення робочої моделі системи

Схема підключення робочої моделі “розумного дому” наведена на рисунку 3.1. Система побудована на основі мікроконтролера NodeMCU v3 LoLin, який містить вбудований Wi-Fi-модуль ESP8266. Попри невисоку вартість, даний контролер є ефективним та зручним інструментом для створення цифрових систем автоматизації, зокрема для проєктів типу “розумний будинок”.

Основні переваги мікроконтролера NodeMCU:

– простота програмування та швидке завантаження прошивки;

– підтримка мови Lua 5.1.4, що забезпечує гнучкість при створенні сценаріїв керування (без режиму налагодження, лише з основними модулями ОС);

– асинхронна подієва модель програмування, що дає змогу ефективно реалізовувати роботу з кількома пристроями одночасно;

– наявність понад 40 вбудованих модулів, які розширюють функціональні можливості контролера;

– прошивка може бути з підтримкою або без підтримки плаваючої точки, причому цілочисельна версія споживає менше пам’яті;

– проєкт активно підтримується спільнотою, а його документація регулярно оновлюється;

– підтримка програмування кількома мовами, зокрема мовою C, що полегшує інтеграцію з іншими системами та бібліотеками.

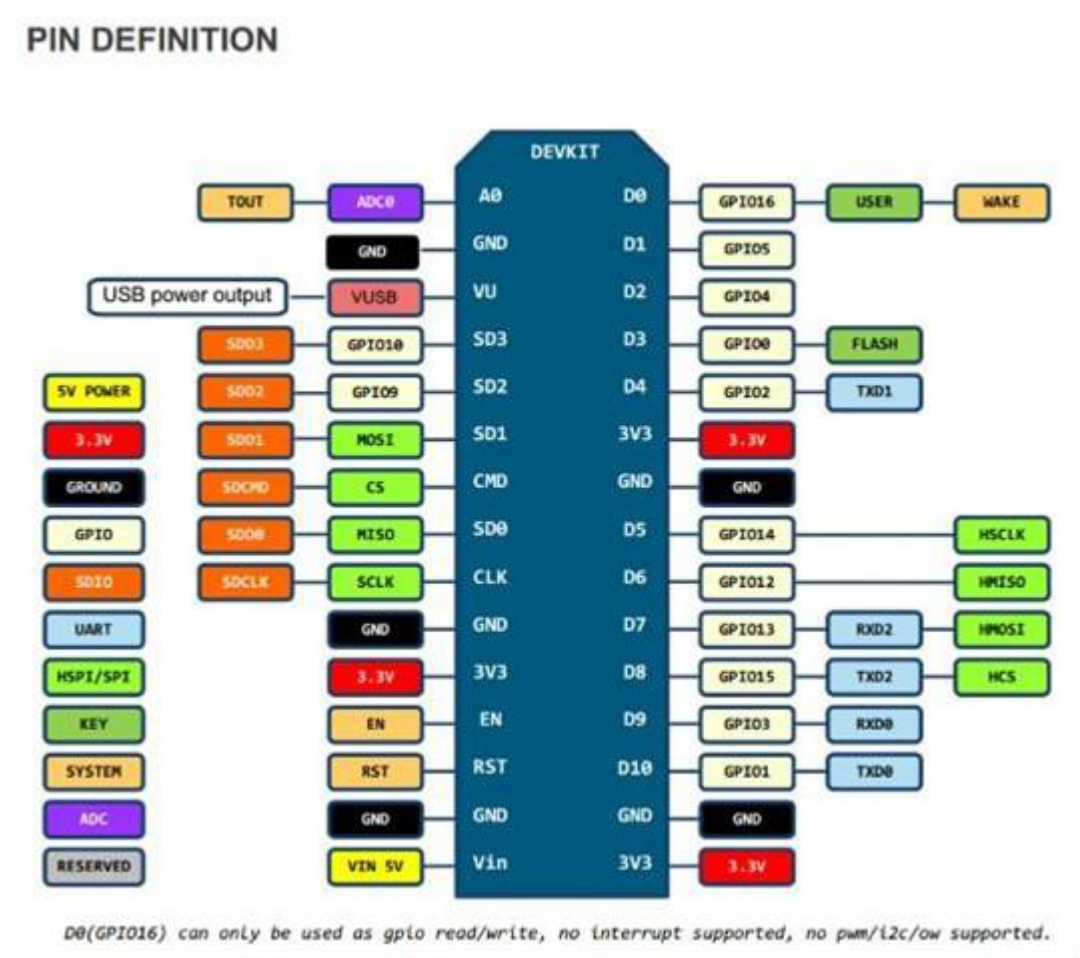

Рисунок 3.1 – Схема мікроконтролера NodeMCU v3 LoLin

Програмування мікроконтролера здійснюється у середовищі розробки Arduino IDE з використанням мови програмування C. Підключення мікроконтролера до персонального комп’ютера виконується через інтерфейс microUSB, який одночасно забезпечує живлення пристрою та передачу даних

під час прошивання. Таким чином, програмування та живлення контролера реалізовані через один інтерфейс, що спрощує процес налагодження системи.

Апаратна конфігурація тестової моделі.

Для тестування розробленої системи до мікроконтролера були підключені такі сенсорні та виконавчі елементи:

– датчик сили/моменту HEX C227986 – призначений для вимірювання механічних навантажень та моментів, що діють на конструктивні елементи системи;

– датчик температури та вологості повітря Aosong AM2302 (DHT22) – забезпечує вимірювання параметрів мікроклімату в приміщенні;

– датчик атмосферного тиску – використовується для моніторингу змін тиску та побудови статистики навколишнього середовища;

– сенсорна панель із чотирма сенсорними кнопками – виконує функції локального керування системою;

– світлодіодна лампа – використовується як виконавчий елемент для візуальної індикації стану або як приклад керованого навантаження.

Схема підключеня показана на рисунку 3.2

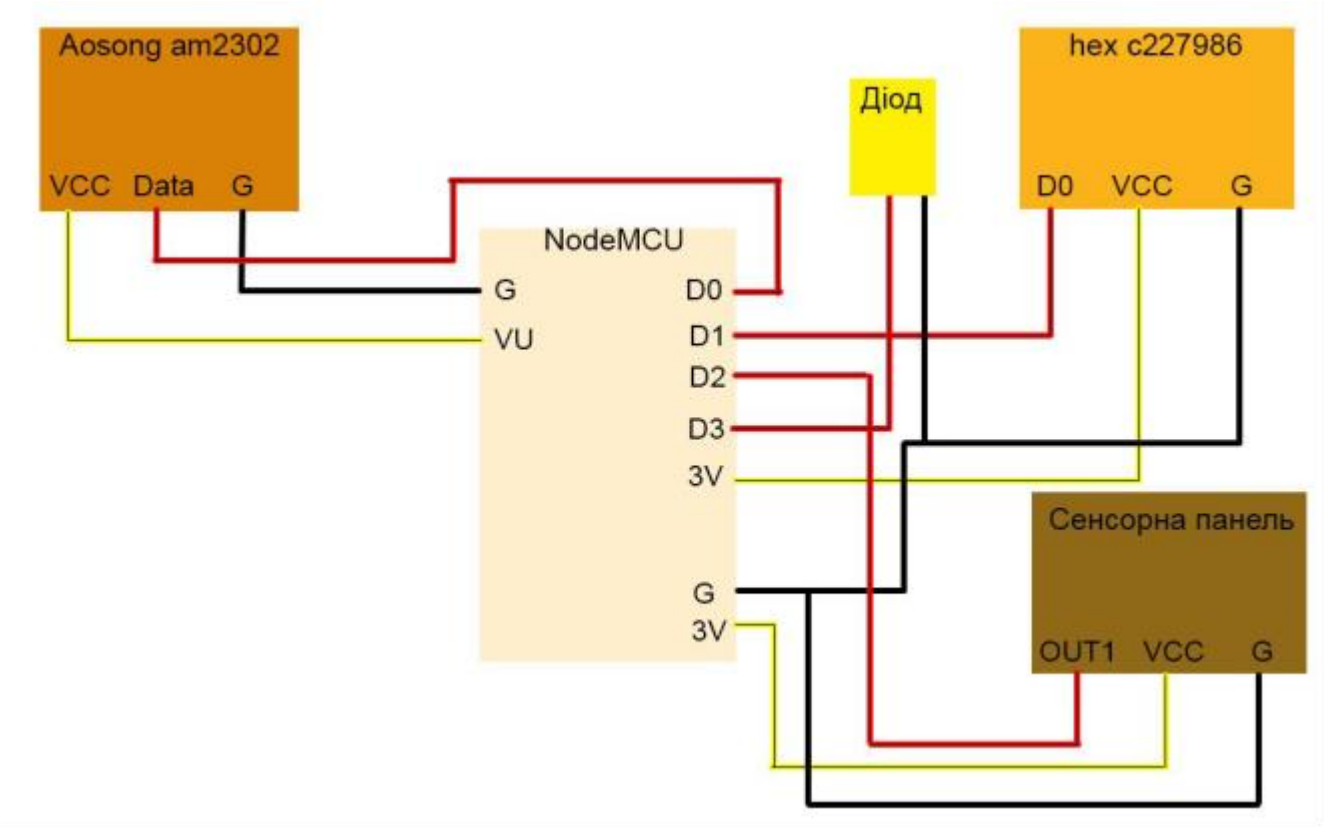

Рисунок 3.2 – Схема підключення робочої моделі

Протокол обміну даними

Для передачі інформації від датчиків у системі було обрано протокол MQTT (Message Queuing Telemetry Transport). MQTT є простим відкритим протоколом обміну повідомленнями, спеціально розробленим для використання в системах Інтернету речей (IoT) та для взаємодії між пристроями з обмеженими ресурсами.

Архітектура MQTT передбачає наявність MQTT-брокера, який виконує роль посередника між клієнтами - так званими MQTT-агентами, що поділяються на видавців (publishers) і передплатників (subscribers). Видавці надсилають (публікують) дані на певні теми (topics), а передплатники отримують лише ті повідомлення, які відповідають обраним темам.

Завдяки своїй легкості, ефективності та надійності навіть за нестабільного мережевого з’єднання, MQTT став одним із основних стандартів для IoT-рішень, у тому числі для систем типу “розумний дім”.

На рисунку 3.3 наведено схему MQTT.

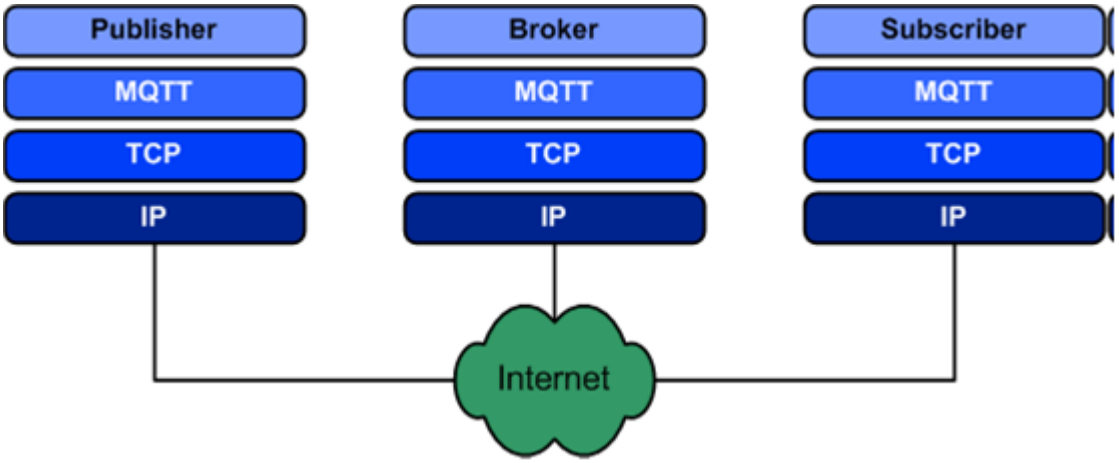

Рисунок 3.3 – Схема MQTT

Принцип роботи протоколу MQTT

Протокол MQTT функціонує за моделлю «видавець – передплатник» (publish–subscribe) та характеризується мінімалістичним набором методів, що визначають дії, необхідні для обміну повідомленнями. Основна взаємодія відбувається між MQTT-агентами та MQTT-брокером, який забезпечує маршрутизацію повідомлень між видавцями та передплатниками.

Агенти встановлюють з’єднання з брокером і далі можуть виконувати одну з двох основних функцій:

– видавці (publishers) – розміщують повідомлення у певних темах (topics);

– передплатники (subscribers) – підписуються на ці теми й отримують повідомлення, опубліковані видавцями.

Після завершення обміну даними агенти можуть відключатися від брокера, завершуючи сеанс зв’язку.

Основні методи протоколу MQTT включають:

– Connect — встановлення з’єднання з MQTT-брокером;

– Disconnect — розрив з’єднання з брокером;

– Publish — публікація повідомлення в обраній темі;

– Subscribe — підписка на певну тему для отримання повідомлень;

– Unsubscribe — відмова від підписки на тему.

У найпростішому вигляді взаємодія між видавцем, брокером та передплатником описується як передача повідомлення через центральний вузол (брокер), який забезпечує фільтрацію, черговість і доставку даних відповідно до підписок клієнтів.

MQTT-брокера наведена на рисунку. 3.4.

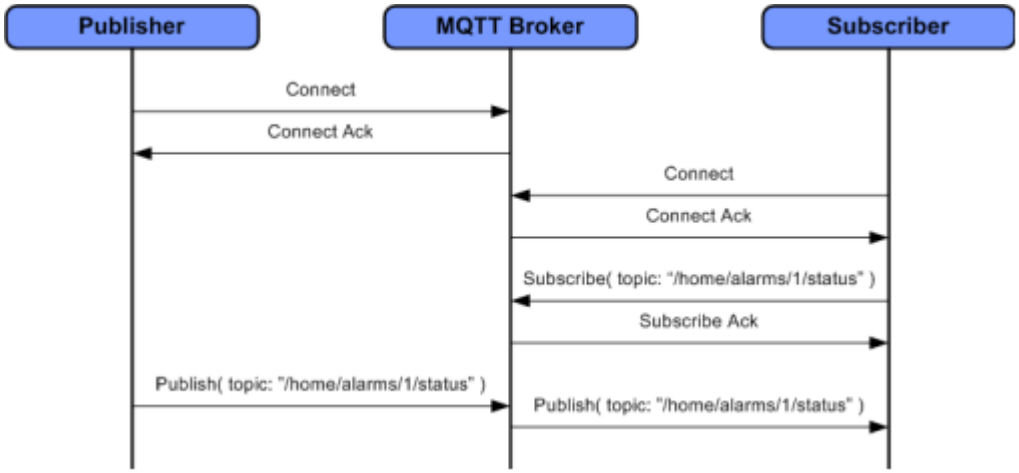

Рисунок 3.4 – Схема роботи MQTT

Рівні якості обслуговування (QoS) у протоколі MQTT

Протокол MQTT підтримує визначення рівня якості обслуговування (Quality of Service, QoS), який задає спосіб і надійність доставки повідомлень між видавцем, брокером і передплатником. Існує три рівні QoS, що відрізняються ступенем гарантії доставки:

– QoS 0 – “At most once” (максимум один раз) Це найпростіший рівень, який реалізує принцип «відправив і забув». Повідомлення надсилається без підтвердження отримання, тому його доставка не гарантується. Такий підхід доцільний для не критичних даних, коли втрата окремих повідомлень не впливає на роботу системи.

– QoS 1 – “At least once” (мінімум один раз) Гарантується, що повідомлення буде доставлено щонайменше один раз. Якщо підтвердження

від приймача не отримано, видавець повторює відправку доти, поки не отримає підтвердження. У результаті можливе дублювання повідомлень, проте втрата даних виключена.

– QoS 2 – “Exactly once” (рівно один раз) Найбільш надійний, але й найповільніший режим доставки. Реалізується чотириступінчаста процедура підтвердження, що гарантує одноразову передачу кожного повідомлення без дублювання. Цей рівень використовується у випадках, коли критично важливо забезпечити цілісність даних.

Вибір конкретного рівня QoS залежить від характеру переданих даних, вимог до надійності зв’язку та допустимої затримки в обміні повідомленнями.

Використання MQTT-брокера CloudMQTT

Для організації обміну повідомленнями в системі було використано хмарний MQTT-брокер CloudMQTT. Обрано тарифний план “Cute Cat” (див. рисунок 3.5), який надає до 10 безкоштовних з’єднань зі швидкістю передачі даних до 10 кбіт/с. Такі параметри є цілком достатніми для тестової моделі системи “розумний дім”, де обсяг даних, що передається сенсорами та виконавчими пристроями, є відносно невеликим.

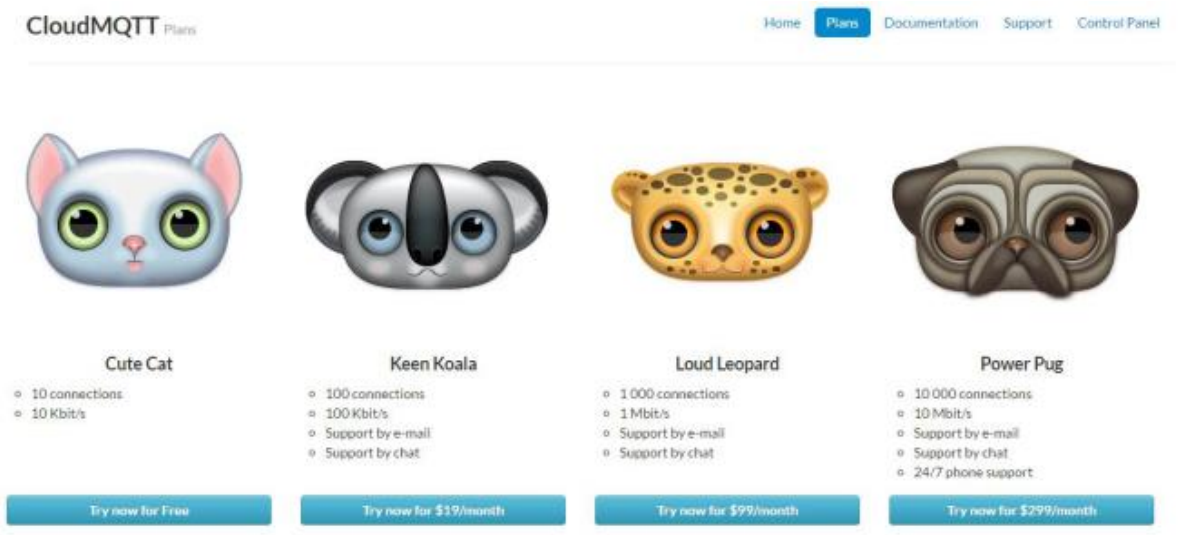

Рисунок 3.5 – Бізнес-плани CloudMqtt

Передача даних від датчиків

Передавання даних від сенсорних пристроїв у системі здійснюється з різною частотою залежно від типу сенсора:

– датчик температури та вологості передає дані один раз на 3 секунди;

– датчик шуму та сенсорна панель передають інформацію один раз на секунду, але лише у випадку зміни показників.

Такий підхід дозволяє оптимізувати обсяг трафіку, зменшити навантаження на канал зв’язку та підвищити загальну стабільність роботи системи.

Веб-застосунок для керування робочою моделлю

Для реалізації керування системою “розумний дім” було створено веб-застосунок, що забезпечує взаємодію користувача з мікроконтролером і хмарною інфраструктурою.

З огляду на невисоке навантаження тестової моделі, прийнято рішення поєднати бізнес-логіку та серверну частину сайту на одному сервері.

У якості серверної платформи обрано Node.js із використанням фреймворка Express, що забезпечує просту структуру, високу швидкодію та зручність інтеграції з фронт-енд частиною, реалізованою мовою JavaScript.

Таке поєднання технологій є логічним вибором, оскільки воно забезпечує швидку розробку, зручну обробку подій у реальному часі та достатню продуктивність для оброблення даних від підключених сенсорів.

Протокол обміну даними між клієнтом і сервером

Через потребу у частому обміні даними між веб-сторінкою та сервером було використано протокол Socket.IO, який забезпечує двосторонній асинхронний обмін даними у режимі реального часу. Це дозволяє системі миттєво реагувати на зміни показників датчиків та оперативно оновлювати інтерфейс користувача без затримок.

3.2 Хмарна інфраструктура

У хмарному середовищі Amazon Web Services (AWS) було розгорнуто віртуальний сервер EC2 під керуванням операційної системи Ubuntu. На ньому встановлено веб-сервер Node.js + Express та розгорнуто базу даних PostgreSQL.

Amazon AWS надає можливість безкоштовного користування протягом року за моделлю IaaS (Infrastructure as a Service), що дозволяє гнучко налаштувати серверне середовище відповідно до потреб системи.

База даних PostgreSQL вибрана завдяки своїй надійності, швидкодії та підтримці складних структур даних, що робить її оптимальним рішенням для зберігання:

– показників, отриманих від сенсорів;
– інформації про користувачів системи;
– історії подій та статистичних даних.

Схема з‘єднання показана на рисунку 3.6

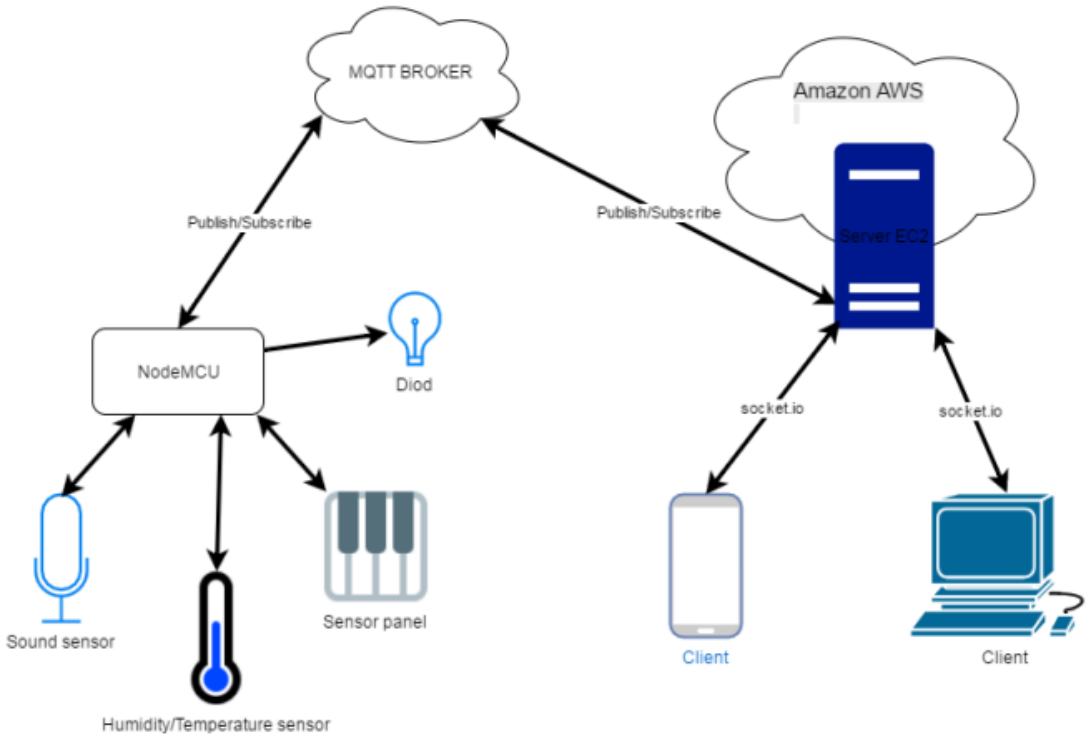

Рисунок 3.6 – Схема зв'язку робочої моделі з хмарним керуванням

Веб-інтерфейс системи

Було розроблено веб-інтерфейс користувача, представлений на рисунку 3.7, який забезпечує моніторинг показників навколишнього середовища та дистанційне керування виконавчими пристроями системи “розумний дім”.

Інтерфейс відображає в реальному часі значення температури та вологості, отримані з відповідних датчиків. Крім того, передбачено візуалізацію рівня звукового шуму з індикацією випадків, коли його рівень перевищує заданий пороговий параметр.

Також реалізовано можливість дистанційного керування світлодіодним індикатором, підключеним до мікроконтролера користувач може вмикати або вимикати його через веб-інтерфейс у режимі реального часу.

Розроблений інтерфейс є інтуїтивно зрозумілим, працює у браузері без необхідності додаткового програмного забезпечення та забезпечує зручну взаємодію користувача із системою моніторингу та керування.

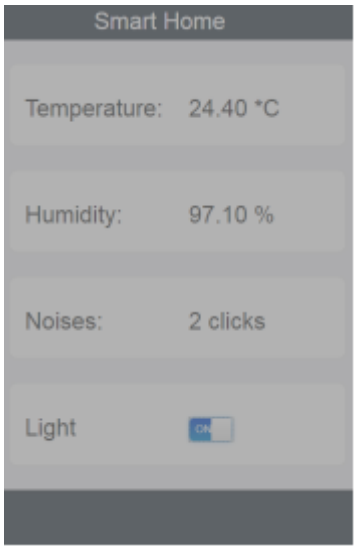

Рисунок 3.7 – Веб-інтерфейс моніторингу даних

Стартова сторінка веб-застосунку представлена на рисунку 3.8 і являє собою вікно авторизації (логіну). Доступ до системи моніторингу даних з датчиків та керування пристроями “розумного дому” є обмеженим.

Усі дії з керування, перегляд статистики або взаємодія з пристроями доступні виключно авторизованим користувачам. Сторонні особи без підтверджених облікових даних не мають можливості отримати доступ до інформації або виконувати будь-які операції з системою.

Такий підхід забезпечує конфіденційність даних, захист від несанкціонованого втручання та відповідає вимогам безпеки при роботі з хмарними системами керування IoT-пристроями.

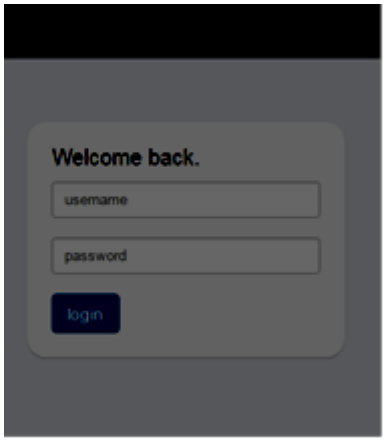

Рисунок 3.8 – Стартове вікно логіну сайта

Після надходження інформації від MQTT-брокера на сервер застосунку, відбувається її попередня обробка. Зокрема, для параметрів температури та вологості реалізовано механізм усереднення значень: після отримання десяти послідовних показників обчислюється їх середнє арифметичне значення.

Отримане усереднене значення надсилається до бази даних у відповідну таблицю разом із часовою міткою (timestamp). е дає змогу вести історію вимірювань, здійснювати подальший аналітичний аналіз та формувати інфографічні моделі змін параметрів навколишнього середовища в часі.

Такий підхід дозволяє зменшити обсяг переданих даних, оптимізувати навантаження на систему та забезпечити точність аналітичних розрахунків.

Висновок

Попри те, що тестова модель системи оперує відносно невеликим набором даних, з точки зору архітектури та побудованої моделі вона має значний потенціал для масштабування. Кожен використаний у ній сенсор є абстрактним представленням більш складного реального пристрою, який може бути без труднощів інтегрований у систему відповідно до потреб користувача.

Під час випробувань було продемонстровано повноцінну двосторонню взаємодію між компонентами системи:

– передавання даних від датчиків до веб-застосунку (датчики → сайт);

– формування керуючих команд від користувача до пристроїв (сайт → датчики).

Важливо зазначити, що розроблена система хмарного керування “розумним домом” є гнучкою та універсальною, і з незначними модифікаціями може бути адаптована до використання інших хмарних платформ та сервісів.

Це стало можливим завдяки застосуванню загальноприйнятих сучасних технологій, зокрема:

– протоколів обміну даними MQTT та Socket.IO;

– мов програмування JavaScript і C;

– мов розмітки HTML та CSS для створення веб-інтерфейсу;

– системи керування базами даних PostgreSQL для збереження та обробки інформації.

Таким чином, тестова модель підтвердила функціональну повноцінність, масштабованість і сумісність розробленої системи з актуальними технологіями хмарного керування у сфері Інтернету речей (IoT).

3.3 Блок-схеми та опис алгоритмів функціонування системи

Після переходу користувача на головну сторінку веб-застосунку, що призначена для онлайн-керування параметрами системи, він має пройти процедуру авторизації. У випадку, якщо користувач не має облікового запису, передбачена можливість реєстрації нового акаунта.

Процес реєстрації включає введення персональних даних (логіну, пароля, адреси електронної пошти тощо), перевірку їх коректності та подальше збереження у базі даних системи. Лише після успішного проходження цієї процедури користувач отримує доступ до функцій моніторингу та керування пристроями “розумного дому”.

На рисунку 3.9 подано блок-схему алгоритму роботи сторінки реєстрації користувача, яка відображає послідовність дій — від введення даних до підтвердження створення нового облікового запису.

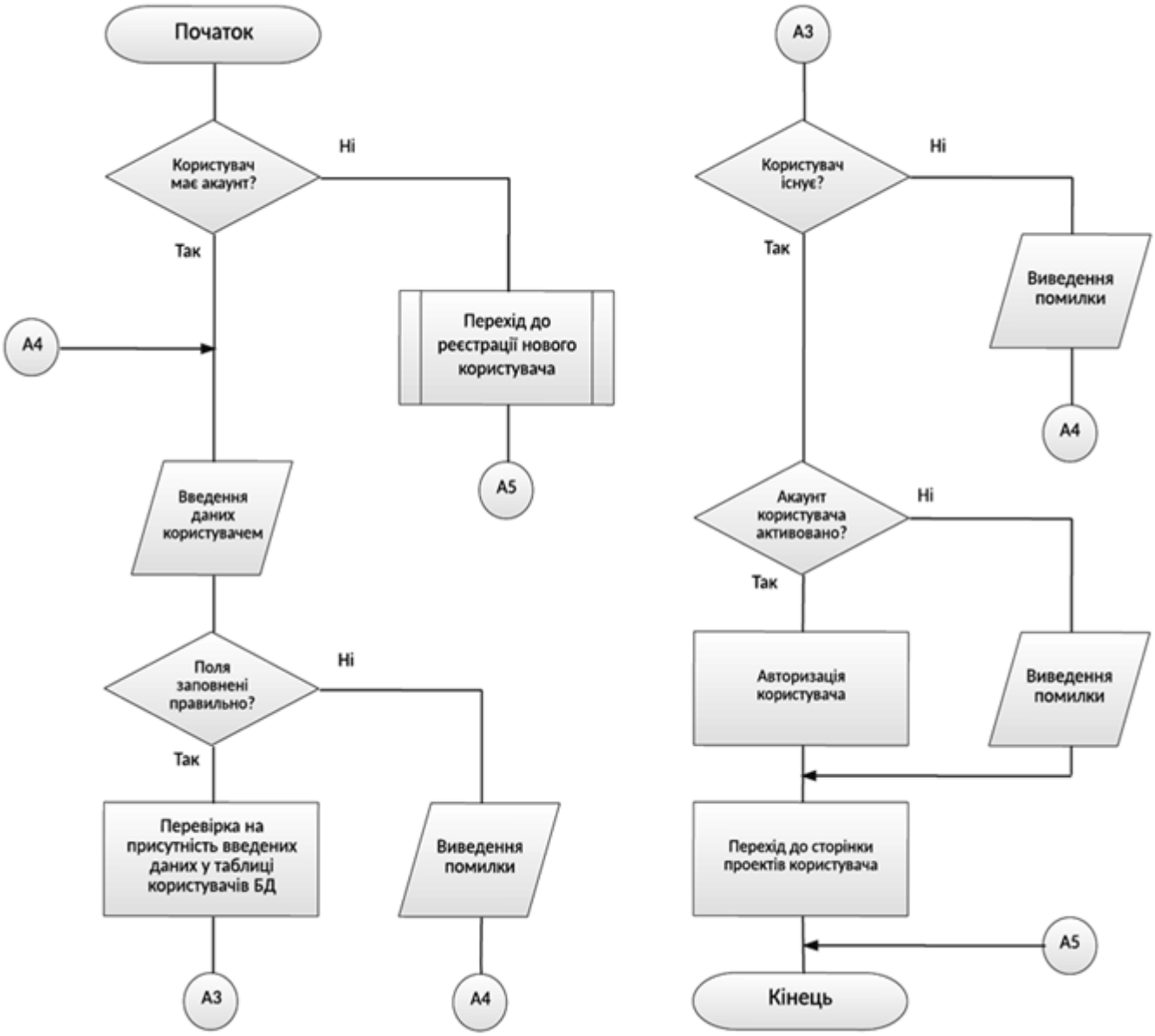

Рисунок 3.9 - блок-схему алгоритму роботи сторінки реєстрації користувача.

Розглянемо алгоритм функціонування програмної частини системи захисту “розумного дому”. Послідовність виконання основних операцій наведено у вигляді блок-схеми, що подана на рисунку 3.10. На схемі відображено основні етапи обробки даних, прийняття рішень та формування керуючих сигналів, які забезпечують роботу підсистеми безпеки у режимі реального часу.

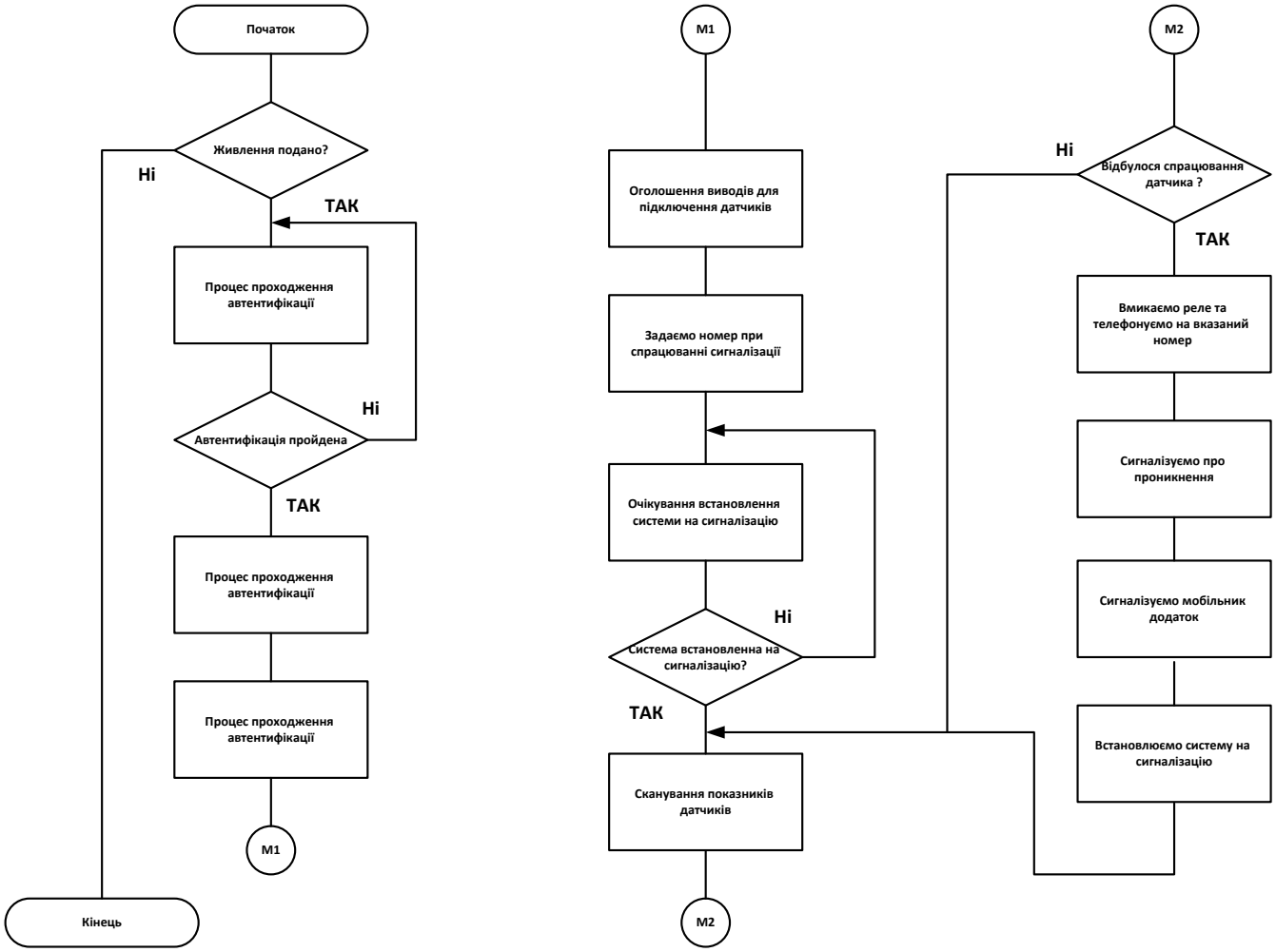

Рисунок 3.10 - Алгоритм роботи програмної частини системи захисту розумного будинку

Як видно з рисунка 3.10, після запуску програми відбувається ініціалізація головного вікна та завантаження основних компонентів. Далі здійснюється підключення бази даних і ініціалізація необхідних бібліотек, що забезпечують функціонування програмного середовища.

На наступному етапі система виконує автоматичне встановлення зв’язку між мікроконтролером і підключеними пристроями. Якщо перевірка зв’язку проходить успішно та отримано відповідний сигнал, програма переходить до сканування всіх підключених приладів і виведення отриманих даних на екран монітора.

Після цього відбувається встановлення параметрів системи за замовчуванням і передача керуючих сигналів до мікроконтролера. У процесі подальшої роботи система постійно аналізує отримані дані; у разі перевищення допустимого рівня споживаної потужності вона автоматично відключає надлишкові прилади відповідно до алгоритму, збереженого в базі даних.

Таким чином, описаний алгоритм відображає один із ключових циклів роботи системи “Розумний дім”. Інші підсистеми функціонують за аналогічним принципом, виконуючи власні алгоритми контролю та керування відповідно до заданих сценаріїв.

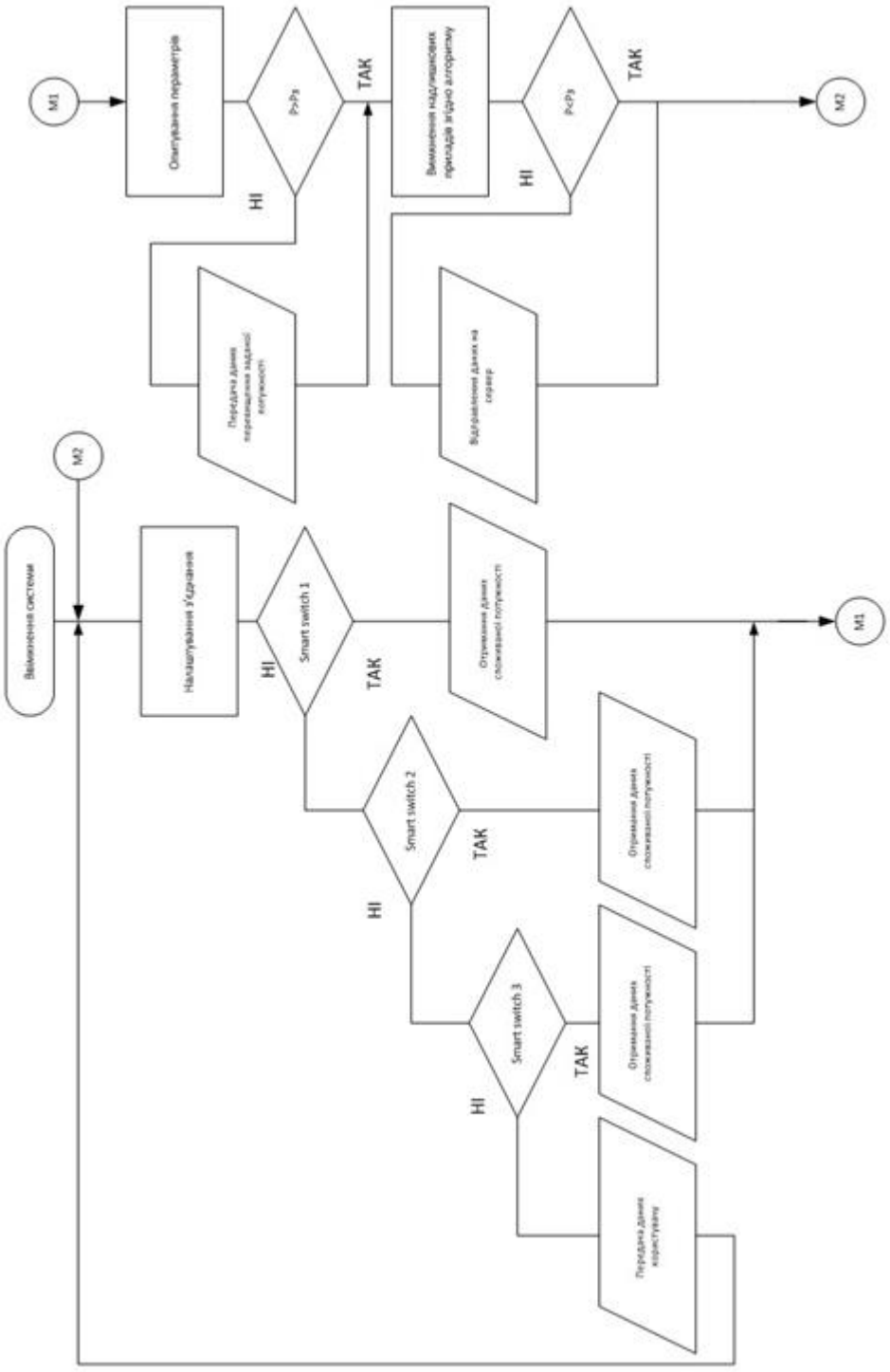

Рисунок 3.11 - блок-схема алгоритму запуску системи.

Алгоритм виконання дії в системі “Розумний дім”

Алгоритм виконання дії в системі “Розумний дім” передбачає послідовність операцій, спрямованих на зміну стану вибраного пристрою відповідно до команди користувача.

На першому етапі здійснюється перевірка доступності фізичного пристрою у мережі. Якщо пристрій недоступний, система формує та виводить повідомлення про відсутність зв’язку з ним. У разі успішного проходження перевірки надсилається запит із параметрами, що змінюють стан пристрою згідно з вибором користувача (наприклад, вмикання, вимикання або регулювання режиму роботи).

Після виконання запиту система отримує зворотний сигнал про результат операції.

Якщо зміна стану пристрою виконана успішно — на екран виводиться повідомлення про успішне виконання дії. У випадку виникнення помилки або відсутності відповіді формується повідомлення про помилку виконання операції.

Послідовність дій алгоритму наведено у вигляді блок-схеми на рисунку 3.12, яка відображає логіку перевірки доступності, надсилання запиту та обробки результатів зміни стану пристрою.

Рисунок 3.12 – Алгоритм операції виконання дії

Алгоритм роботи планувальника дій

Алгоритм планувальника дій у системі «Розумний дім» реалізовано у вигляді циклічного процесу, що постійно контролює настання заданого часу виконання певної дії.

Під час роботи планувальник перевіряє поточний системний час і порівнює його з часом, встановленим у розкладі завдань. Якщо умова збігу часу виконується - система здійснює заплановану дію (наприклад, вмикання освітлення, активація поливу або запуск кліматичної системи). Якщо запланований час ще не настав, програма переходить у режим очікування, після чого повторно виконує перевірку.

Такий підхід забезпечує автоматичне керування пристроями відповідно до заздалегідь заданих сценаріїв і сприяє оптимізації роботи системи.

Послідовність дій планувальника відображено у вигляді блок-схеми на рисунку 3.13, яка демонструє логіку перевірки часу, виконання дії та повторного циклу очікування.

Рисунок 3.13 – Алгоритм роботи планувальника задач

Алгоритм роботи сервісу моніторингу системи

Алгоритм моніторингу стану системи “Розумний дім” забезпечує контроль доступності пристроїв і отримання актуальної інформації про їхній поточний стан.

На початковому етапі здійснюється перевірка доступності вибраного фізичного пристрою у мережі. Якщо пристрій недоступний, система формує та виводить повідомлення про його недоступність для інформування користувача. У разі успішного проходження перевірки надсилається запит на отримання параметрів стану обраного пристрою згідно з дією, ініційованою користувачем.

Отримана інформація відображається у веб-інтерфейсі системи або передається до бази даних для подальшої обробки. Таким чином, алгоритм забезпечує постійний контроль працездатності обладнання, зворотний зв’язок із користувачем і підвищує надійність роботи системи в цілому.

Послідовність виконання операцій моніторингу наведено у вигляді блок-схеми на рисунку 3.14, де показано процес перевірки доступності, формування запиту та отримання поточного стану пристрою.

Рисунок 3.14 – Алгоритм операції виконання дії

Блок схема основного алгоритма представлено на рисунку 3.15

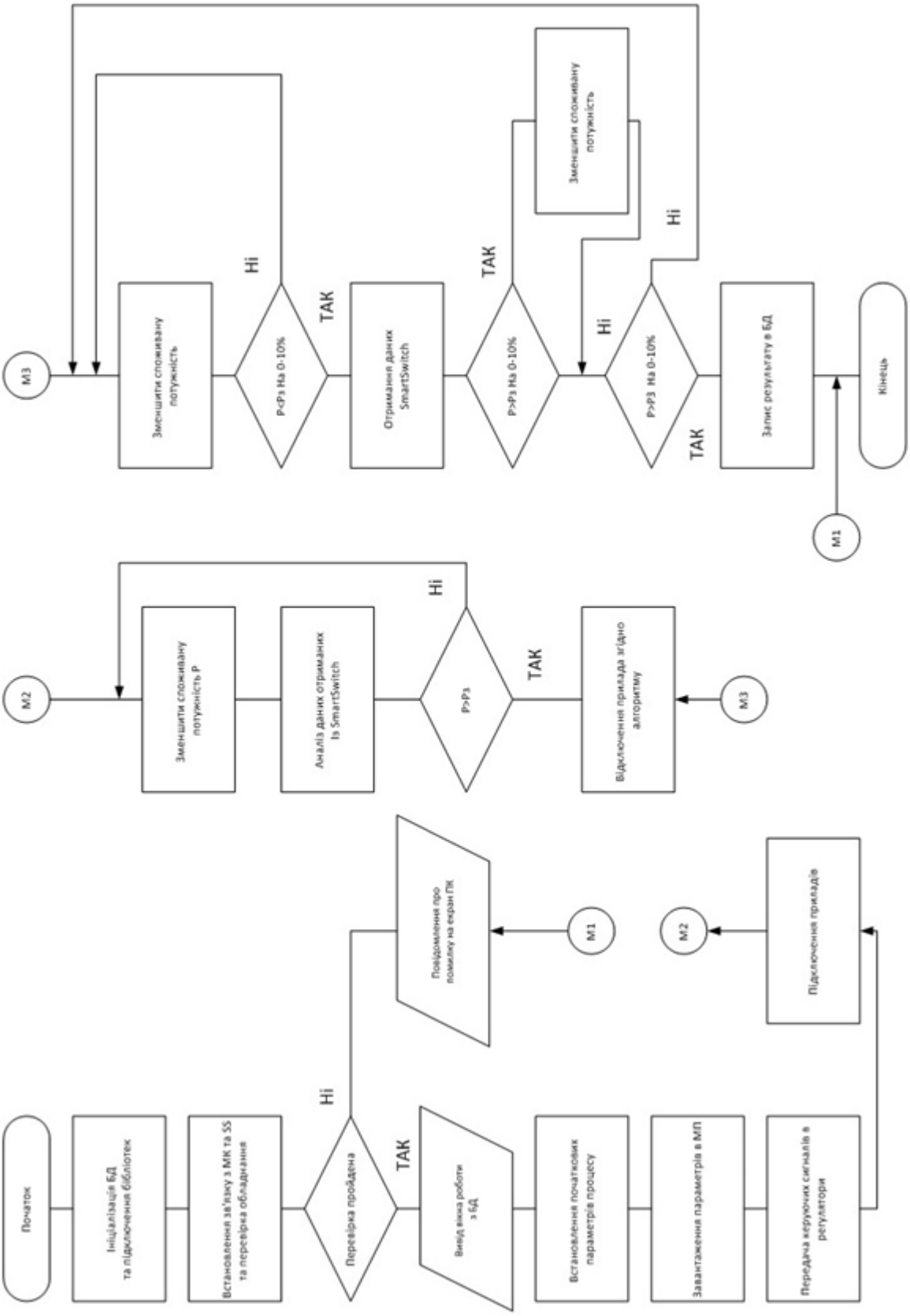

Рисунок 3.15 - блок-схема основного алгоритму роботи системи

3.4 Захист інформації в системі

Методи захисту інформації в системі «Розумний дім»

Забезпечення захисту інформації в системах Інтернету речей (IoT) може здійснюватися кількома способами. Один із підходів полягає у кодуванні та декодуванні даних безпосередньо на IoT-пристроях. Проте, через обмежену обчислювальну потужність мікроконтролерів, реалізація складних криптографічних алгоритмів може призвести до зниження швидкодії системи.

Альтернативним варіантом є перенесення функцій безпеки та надійності передачі даних на рівень комп'ютерної або хмарної мережі. У цьому випадку передбачається, що мережеве середовище, у якому працюють пристрої, є ізольованим та захищеним. Для організації безпечного віддаленого підключення можуть використовуватися VPN-тунелі або SSH-з'єднання, які забезпечують шифрування переданих даних і контроль автентичності сторін.

Захист бездротової мережі Wi-Fi

Технологія Wi-Fi є одним із найпоширеніших способів підключення пристроїв IoT до локальної мережі. За умови правильного налаштування вона забезпечує достатній рівень безпеки, проте потребує відключення вразливих режимів, таких як WEP та QSS, які легко піддаються несанкціонованому доступу.

Найпоширенішим сучасним стандартом шифрування є WPA2-PSK, що використовує алгоритм AES. Відповідно до досліджень [26], злам точки доступу з таким типом шифрування можливий лише за допомогою повного перебору паролів. Так, словник із 250 мільйонів комбінацій (приблизно 2 ГБ) потребує понад 60 годин для повного перебору на звичайному ноутбуці, а для паролів із дев'яти символів обсяг словника зростає до 900 мільйонів записів, що вимагає кількох тижнів безперервної роботи. Отже, використання складних паролів і надійних алгоритмів шифрування є обов'язковою умовою безпечної експлуатації Wi-Fi-мережі.

Аутентифікація у протоколах передачі даних

Більшість протоколів обміну інформацією підтримують механізми автентифікації, проте вони часто залишаються вимкненими через спрощення конфігурації системи. Це є помилковим підходом, оскільки навіть базові елементи безпеки значно ускладнюють можливість несанкціонованого доступу.

Протоколи MQTT, REST та SNMP мають вбудовану підтримку автентифікації користувачів, яку можна використовувати для захисту переданих даних і керування пристроями. Застосування цих функцій підвищує рівень безпеки системи без необхідності істотних апаратних змін.

Модель підключення через VPN або SSH

VPN і SSH є найпоширенішими засобами побудови захищених тунелів передачі даних. При використанні VPN необхідно налаштувати маршрутизатор або міні-комп’ютер, який здійснює маршрутизацію та шифрування трафіку. У випадку SSH-тунелювання дані передаються через захищений мережевий протокол SSH, який вимагає наявності пристрою, що забезпечує переадресацію портів і шифрування сеансу.

Обидва варіанти особливо ефективні при використанні міні-комп’ютерів з ОС Linux, які підтримують необхідні інструменти шифрування та аутентифікації. Як зазначено в [27], одним із практичних методів захисту SSH-з’єднань є обмеження кількості невдалих спроб підключення, що значно ускладнює здійснення атак перебором паролів.

У розробленій системі передбачено можливість створення VPN або SSH-тунелю між міні-комп’ютером та централізованим сервером. Датчики, побудовані на базі мікроконтролерів, підключаються до сервера безпосередньо через MQTT або за допомогою локального MQTT-брокера, який приймає повідомлення у своїй мережі та пересилає їх до центрального вузла.

Отже, для підвищення надійності та безпеки системи доцільно використовувати стандартні засоби протоколів зв’язку та шифрування. Слід

керуватися принципом, що безпеки не буває забагато: необхідно уникати простих паролів, активувати механізми автентифікації у всіх доступних протоколах і використовувати сучасні методи шифрування трафіку для захисту даних користувача.

Захист програмного забезпечення.

Для забезпечення захисту даних у розробленому програмному забезпеченні веб-системи командного керування проєктами було використано вбудовані механізми безпеки мови програмування PHP. Ці функції дозволяють запобігати несанкціонованому доступу, захищати дані користувачів та мінімізувати ризики уразливостей, пов’язаних із роботою веб-додатка. password_hash() і password_verify().

Password_hash()

Захист даних у веб-системі

Для забезпечення захисту користувацьких даних у розробленій веб-системі командного керування проєктами використано вбудовані функції безпеки мови PHP, які дозволяють запобігти несанкціонованому доступу, викраденню паролів і поширеним типам атак.

Хешування паролів користувачів

Для захисту облікових даних застосовуються функції password_hash() та password_verify():

– password_hash() — використовується для створення хешу введеного користувачем пароля;

– password_verify() — перевіряє, чи відповідає введений пароль збереженому хешу.

Функція password_hash() приймає два основні параметри:

1. Перший параметр — пароль, який необхідно захистити;
2. Другий параметр — алгоритм хешування.

Підтримуються два варіанти алгоритмів:

– PASSWORD_DEFAULT — використовує алгоритм BCrypt, який формує хеш довжиною 60 або більше символів;

– PASSWORD_BCRYPT — базується на алгоритмі CRYPT_BLOWFISH і завжди генерує хеш рядком із 60 символів.

Перевагою функції password_hash() є те, що вона автоматично генерує “сіль” і самостійно керує складністю (вартістю) обчислення хешу, що зменшує ймовірність помилок під час реалізації безпеки. За потреби програміст може задати власні параметри, наприклад значення вартості обчислення або “солі”, передавши їх як третій аргумент функції.

Приклад використання:

```
<?php
echo password_hash("rasmuslerdorf", PASSWORD_DEFAULT);
?>
```

Результатом виконання функції може бути рядок: $2y$10$.vGA1O9wmRjrwAVXD98HNOgsNpDczlqm3Jq7KnEd1rVAGv3Fykk1a

Приклад із власними параметрами:

```
<?php
$options = ['cost' => 12];
echo password_hash("rasmuslerdorf", PASSWORD_BCRYPT, $options);
?>
```

Результат: $2y$12$QjSH496pcT5CEbzjD/vtVeH03tfHKFy36d4J0Ltp3lRtee9HDxY3K

Як видно, обидва хеші відрізняються, що підтверджує унікальність “солі” та додаткову стійкість до атак методом перебору.

Захист від XSS-атак

Для запобігання виконанню впроваджених JavaScript-скриптів (XSS-атак) застосовано стандартні функції PHP:

– htmlspecialchars() — замінює спеціальні символи HTML, зокрема < та >, на безпечні еквіваленти < і >;

– strip_tags() — видаляє з рядка всі HTML-теги, залишаючи лише текстовий вміст.

Крім того, у файлі .htaccess додано фільтрацію запитів із потенційно шкідливими параметрами:

```
Options +FollowSymLinks
RewriteEngine On
RewriteCond %{QUERY_STRING} (\<|%3C).*script.*(\>|%3E) [NC,OR]
RewriteCond %{QUERY_STRING} GLOBALS(=|\[|\%[0-9A-Z]{0,2}) [OR]
RewriteCond %{QUERY_STRING} _REQUEST(=|\[|\%[0-9A-Z]{0,2})
RewriteRule ^(.*)$ index.php [F,L]
```

Цей код блокує спроби виконання скриптів через рядок запиту, зменшуючи ризик XSS-атак на сайт.

Захист від SQL-ін’єкцій

Для попередження SQL-ін’єкцій, що полягають у впровадженні шкідливих SQL-команд у запити до бази даних, використовується функція mysql_real_escape_string().

Вона екранує спеціальні символи у рядках, які передаються в SQL-запит, враховуючи кодування з’єднання з базою даних.

Таким чином, навіть якщо користувач спробує вставити у форму запиту фрагмент SQL-коду, він буде сприйнятий як звичайний текст, а не як частина інструкції запиту.

Застосування описаних функцій та механізмів у поєднанні з правильною структурою коду забезпечує високий рівень безпеки веб-системи, захист автентифікаційних даних користувачів і стійкість до основних типів атак (XSS, SQL-ін’єкцій, підбору паролів).

4 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

4.1 Реєстрація доменних імен

Реєстрація домену — це процес внесення до реєстру відповідної доменної зони запису про нове доменне ім’я. Процедура реєстрації є досить простою:

– необхідно створити обліковий запис у реєстратора доменних імен;
– поповнити рахунок користувача;
– перевірити обране доменне ім’я на наявність або зайнятість;
– у разі його доступності — подати заявку на реєстрацію.

Після внесення запису до реєстру, який містить дані про адміністратора домену, реєстратора, дату реєстрації, термін дії та статус делегування, доменне ім’я стає активним протягом 5–10 хвилин.

Для коректного функціонування домену необхідно вказати DNS-сервери (делегувати домен) у налаштуваннях облікового запису реєстратора, що забезпечить маршрутизацію запитів до хостинг-сервера.

Реєстратор доменних імен — це організація, уповноважена створювати (реєструвати) нові доменні імена або продовжувати термін дії вже існуючих.

Реєстрація обов’язкова для таких типів доменів:

– домен нульового рівня (кореневий);
– усі домени верхнього рівня (першого рівня);
– деякі домени другого рівня, наприклад com.ru або co.uk.

Для створення субдоменів у межах інших доменів спеціальні повноваження, як правило, не потрібні.

Хостинг

Хостинг (від англ. hosting) — це послуга, що передбачає надання дискового простору, мережевих ресурсів та технічної інфраструктури для

розміщення вебресурсу на сервері, який постійно перебуває в мережі (наприклад, в Інтернеті).

Хостинг охоплює широкий спектр послуг, що залежать від типу апаратного та програмного забезпечення. У типовому випадку під хостингом розуміють розміщення файлів вебсайту на сервері, який має встановлене необхідне програмне забезпечення (вебсервер, СУБД, скриптові модулі тощо).

Зазвичай до пакету хостингових послуг входить:

– простір для вебфайлів і баз даних;

– підтримка електронної пошти;

– DNS-сервіс;

– сховище файлів;

– супутні програмні сервіси.

– У разі потреби ці послуги можуть надаватися окремо, наприклад:

– поштовий хостинг — розміщення поштових серверів і кореспонденції;

– файловий хостинг — зберігання та передавання клієнтських файлів;

– відеохостинг — зберігання та трансляція відеофайлів.

Хостинг-провайдери часто пропонують комплексні рішення, що включають реєстрацію домену, створення вебсайту та надання додаткового програмного забезпечення.

Послуги можуть надаватися як спеціалізованими хостинговими компаніями, так і великими провайдерами інформаційних сервісів, такими як Google, Microsoft, Yahoo тощо.

Розрізняють безкоштовний і платний хостинг. Безкоштовні провайдери зазвичай фінансують свою діяльність через розміщення реклами або надання додаткових платних сервісів у поєднанні з базовими безкоштовними.

Платний хостинг, своєю чергою, забезпечує вищу стабільність роботи, технічну підтримку, захист даних і більші ресурси, що робить його доцільним вибором для професійних вебсистем.

4.2 Завантаження вихідного коду

Після придбання доменного імені та послуг по хостингу необхідно завантажити на сервер вихідний код розробленого ПЗ за допомогою програм на кшталт FileZilla та перевірити правильність роботи сайту.

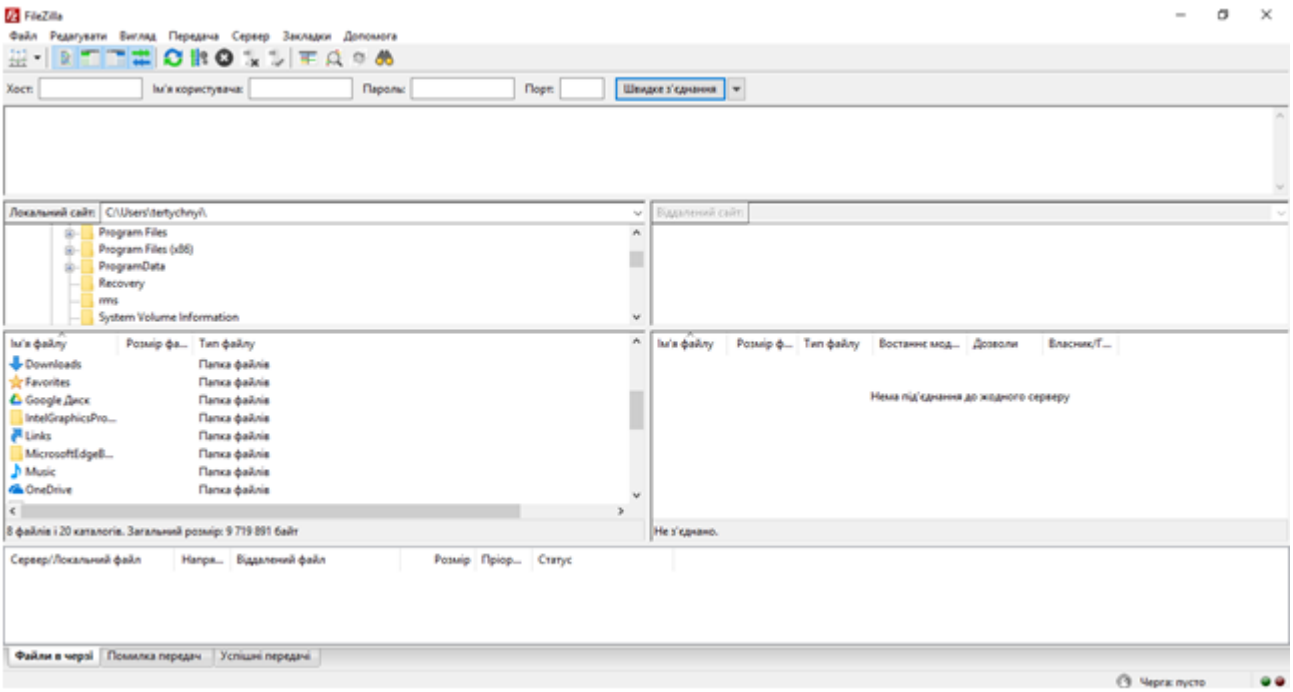

Рисунок 4.1 – Головне вікно програми для завантаження файлів на сервер

Після придбання доменного імені та замовлення послуг хостингу необхідно завантажити вихідний код розробленого програмного забезпечення на сервер.

Це здійснюється за допомогою FTP-клієнтів, зокрема таких програм, як FileZilla, WinSCP або Cyberduck.

За допомогою даних програм користувач має можливість підключитися до серверу, скопіювати файли веб-додатка у відповідний каталог хостингу та перевірити коректність роботи сайту після завантаження.

Керування базою даних

Після розміщення файлів сайту на сервері здійснюється налаштування та завантаження бази даних.

– для адміністрування СУБД MySQL або MariaDB було використано веб-додаток phpMyAdmin — програму з відкритим кодом, написану мовою PHP, що має зручний графічний веб-інтерфейс.

Даний інструмент дозволяє:

– виконувати адміністрування серверів баз даних через веб-браузер;

– запускати SQL-запити;

– переглядати, змінювати або видаляти вміст таблиць;

– експортувати та імпортувати структуру й дані бази.

Популярність phpMyAdmin пояснюється його простотою використання, відсутністю потреби у встановленні додаткового ПЗ та можливістю керування базами даних із будь-якого пристрою, підключеного до Інтернету.

Більшість українських хостинг-провайдерів пропонують phpMyAdmin як стандартну панель керування базами даних, що підтверджує його практичну цінність та надійність.

Реалізація та запуск системи управління «Розумний дім»

Розроблена система «Розумний дім» побудована за модульним принципом, що передбачає розділення функціоналу на незалежні компоненти.

Така архітектура зменшує взаємозалежність між модулями, полегшує модернізацію системи та підвищує можливість повторного використання реалізованих рішень.

Користувацький інтерфейс має інтуїтивно зрозумілу структуру, що забезпечує низький поріг входу для нових користувачів.

Інтерфейс системи тестування створено з метою максимального спрощення взаємодії, забезпечуючи швидкий доступ до основних функцій контролю й керування.

У роботі подано опис основних компонентів програмного забезпечення, а також детальний опис функціональних можливостей.

Розроблено користувацьку інструкцію, що демонструє послідовність дій при роботі з додатком.

– послідовність роботи користувача

– після запуску програми з’являється вікно завантаження, після чого відкривається форма авторизації.

– користувач вводить логін і пароль.

Якщо дані некоректні — система виводить повідомлення про помилку та пропонує повторити спробу натисканням кнопки «TRY AGAIN».

У разі успішної авторизації відкривається головне меню системи з вибором режимів роботи.

Основні режими:

- Моніторинг системи — відкриває вікно для перегляду поточного стану підключених пристроїв у реальному часі.

- Операції з системою — дозволяє змінювати параметри роботи пристроїв.

Після вибору пристрою користувачу відображається його стан і дві кнопки:

– змінити параметри або повернутись у попереднє меню.

– наприклад, при зміні температури використовується елемент управління spinner, після вибору значення слід натиснути «ACCEPT».

– планувальник завдань - дозволяє створювати розклад для автоматичного виконання операцій.

– користувач задає час активації дії, натискає «OK» для підтвердження або «Cancel» для скасування.

– переглядати заплановані події можна свайпом вправо, а редагувати або видаляти - за допомогою відповідних жестів і контекстного меню.

– повернення до головного меню здійснюється свайпом вліво.

У процесі реалізації та тестування системи всі вимоги, визначені на етапі технічного завдання, були повністю виконані.

Розроблена система успішно функціонує, забезпечує моніторинг, управління та планування подій у межах комплексу «Розумний дім», а її архітектура дозволяє масштабування та подальше вдосконалення.

ВИСНОВКИ

У результаті виконання магістерської роботи створено систему хмарного управління розумним будинком із реалізованою підсистемою безпеки передавання даних.

В межах України подібні вітчизняні розробки у цій сфері представлені недостатньо, що підкреслює актуальність теми дослідження.

У магістерській роботі наведено теоретичне узагальнення та вирішено наукове завдання, пов’язане з дослідженням методів управління розумним будинком із впровадженням системи хмарного керування.

Рішення поставленого наукового завдання полягало у виконанні таких основних етапів:

– проведено аналіз і порівняльний огляд існуючих систем управління розумним будинком із підсистемами безпеки передавання даних;

– досліджено принципи побудови та функціонування системи управління розумним будинком із впровадженням підсистеми хмарного управління;

– на основі отриманих результатів розроблено програмну реалізацію системи управління розумним будинком, що забезпечує безпечну передачу даних та підвищений рівень кіберзахисту.

Розроблені в ході виконання магістерської роботи алгоритми забезпечують ефективне вирішення завдань хмарного управління системою «Розумний дім» та підтверджують працездатність запропонованих технічних і програмних рішень.

Проведено аналіз предметної області, у ході якого визначено основні об’єкти, взаємодія між якими має ключове значення для функціонування системи, а також встановлено їхні основні характеристики та взаємозв’язки. На основі результатів аналізу було побудовано алгоритм роботи системи та обрано оптимальне середовище розробки програмного забезпечення.

Розроблена система характеризується простим, інтуїтивно зрозумілим і зручним інтерфейсом користувача, що забезпечує легкість у освоєнні, комфорт у використанні та не вимагає спеціальної технічної підготовки.

Розроблена програма реалізована мовами високого рівня C++ та JavaScript, що забезпечують ефективну обробку даних і стабільну роботу системи. Використання цих мов дозволило оптимізувати процес розроблення програмного забезпечення, скоротивши терміни створення та, відповідно, зменшивши витрати на його реалізацію.

Надано необхідні рекомендації щодо встановлення та налаштування розробленого програмного забезпечення.

Для підвищення рівня інформаційної безпеки в системі запропоновано використовувати змішану схему шифрування, що поєднує алгоритми Base64, AES та RSA. Такий підхід забезпечує оптимальне співвідношення між швидкодією, надійністю та стійкістю до кібератак.

Розроблений продукт має потенціал для подальшого вдосконалення та може бути адаптований для використання в інших галузях, де необхідні системи автоматизованого управління з підвищеним рівнем захисту даних.

СПИСОК ЛІТЕРАТУРИ

1. Саліхов М.М. Самокеровані автомобілі та системи їх навігації// Навч. посібник / В. Є. Бахрушин. – Запоріжжя: КПУ, 2011. – 268 с

2. Кузнецов Ю.М., Луців І.В., Дубиняк С.Г. Теорія технічних систем. -К.: Тернопіль, 1998.-310с.

3. Стеклов В.К. Проектування систем автоматичного керування. - К.:Вища школа,1995.-231 с.

4. Мигаль В. Д. Інтелектуальні системи в технічній експлуатації автомобілів: монографія. Х.: Майдан, 2018. 262 с.

5. Романенко В.Д. Методи автоматизації прогресивних технологій.- К.:Вища школа,1995.-519 с.

6. Рудик А. В. Наукові основи та принципи побудови приладової системи.

7. Koval V., Adamiv O., Proc. of the Third IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2005). – Sofia (Bulgaria). – 2005. – P. 120- 124.

8. Зайцев Г.Ф., Стеклов В.К., Бріцький О.І. Теорія автоматичного управління. – К.: Техніка, 2002. – 688 с.

9. Довідник по автоматизації с/г виробництва //За ред. І.І. Мартиненка.-К.:Урожай,1985.-212 с.

10. Мигаль В. Д. Інтелектуальні системи в технічній експлуатації автомобілів: монографія. Х.: Майдан, 2018. 262 с.

11. Попович М.Г., Ковальчук О.В. Теорія автоматичного керування: Підручник. – 2-ге вид., – К.: Либідь, 2007. - 656 с.

12. Автоматизація технологічних процесів і виробництв харчової промисловості: Підручник/ Ладанюк А.П.,Трегуб В.Г., Ельперін І.В., Цюцюра В.Д. – К.: Аграрна освіта, 2001 – 224 с.

13. Навігаційні системи [Електронний ресурс] : навч. посіб. для студ. Спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» / С.Л. Лакоза; КПІ ім. Ігоря Сікорського, 2021. — 80 с

14. Simulink Documentation [Електронний ресурс]. - Режим доступу: http://www.mathworks.com/ access/ helpdesk/ help/ toolbox/ simulink.

15. Жидецький В. Ц. Основи охорони праці : підруч. Львів : Афіша, 2005. 350 с.

16. Гогіташвілі Г. Г., Лапін В. М. Основи охорони праці : навч. посіб. 3-є вид., стереотипн. Львів : «Новий Світ – 2000». 2006. 232 с.

17. Босов Є. П., Жесан Р. В., Каліч В. М., Голик О. П., Зубенко В. О. Охорона праці при проектуванні систем автоматизації виробництва : навч. посіб. 2-е вид., перероб. і доп. Кропивницький : ЦНТУ, 2022. 208 с.

18. Конституція України. Київ : Андронум, 2020. 60 с.

19. Про охорону праці : Закон України. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text (дата звернення: 21.10.2024).

20. Основи законодавства України про охорону здоров’я : Закон України. URL: https://zakon.rada.gov.ua/laws/show/2801-12#Text (дата звернення 03.11.2024).

21. Про систему громадського здоров’я : Закон України. URL: https://zakon.rada.gov.ua/laws/show/2573-20#n840 (дата звернення 03.11.2024).

22. Про використання ядерної енергії та радіаційну безпеку : Закон України. URL: https://zakon.rada.gov.ua/laws/show/39/95-%D0%B2%D1%80 (дата звернення 29.10.2024).

23. Про загальнообов’язкове державне соціальне страхування : Закон України. URL: https://zakon.rada.gov.ua/laws/show/1105-14 (дата звернення 24.10.2024).

24. Кодекс цивільного захисту України. URL: https://zakon.rada.gov.ua/laws/main/5403-17#Text (дата звернення: 17.11.2024).

25. Кодекс законів про працю України. URL: https://zakon.rada.gov.ua/laws/main/322-08#Text (дата звернення: 07.10.2024).

26. Правила улаштування електроустановок : вид. офіц. Київ : Міненерговугілля України, 2017. 617 с.

27. ВікіпедіЯ. Вільна енциклопедія : веб-сайт. URL: https://uk.wikipedia.org/wiki/ (дата звернення: 31.09.2024).

28. Жидецький В. Ц., Джигирей В. С., Сторожук В. М., Туряб Л. В., Лико Х. І. Практикум з охорони праці. Львів : Афіша, 2000. 352 с.

29. Іванов В. Г., Дзюндзюк Б. В., Олександров Ю. М. Охорона праці в електроустановках : навч. посіб. / за ред. В. Г. Іванова. Київ : Око, 1994. 226 с.