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

ТРАЧ Мар'ян Богданович

Програмно керована модульна архітектура IoT / Software-driven modular IoT architecture

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

Кваліфікаційна робота

Виконав студент групи КН-43
М. Б. Трач

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

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

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

Тернопіль – 2025

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

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ТРАЧ Мар'ян Богданович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Програмно керована модульна архітектура IoT / Software-driven modular IoT architecture
керівник роботи О.Р. Осолінський, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- проаналізувати існуючі програмно керовані архітектури;
- провести аналіз існуючих рішень;
- сформувати постановку задачі;
- розробити концепцію багаторівневої програмно керованої архітектури IoT;
- розробити функціональну модель шлюзового модуля, яка включає алгоритм протокольної нормалізації, буферизацію, мультипротокольну взаємодію та автоматичне визначення типу трафіку;
- реалізувати механізм взаємодію шлюзу з рівнями системи;
- обґрунтувати вибір алгоритмів для вибору локального контролера в динамічних мережах;
- реалізувати мінімальну програмну реалізацію частини шлюзового модуля з протокольною нормалізацією;
- провести тестування шлюзового модуля з протокольною нормалізацією.
5. Перелік графічного матеріалу в роботі:
- реалізована архітектура;
- основні алгоритми функціонування.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

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

Студент_____ М. Б. Трач
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмно керована модульна архітектура IoT» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 56 сторінок і містить 17 ілюстрацій, 3 додатки та 46 використаних джерел.

Метою кваліфікаційної роботи є розробка програмно керованої модульної архітектури IoT з використанням підходів програмно-конфігурованих мереж (SDN), яка забезпечує протокольну нормалізацію, збереження даних, інтеграцію з сенсорною інфраструктурою та взаємодію з модулями аналітики і візуалізації.

В роботі застосовано методи аналізу архітектур і алгоритмів для систем Інтернету речей, моделювання структури шлюзових модулів, розробки алгоритмів нормалізації даних, засоби системного програмування, а також інструменти візуалізації та тестування прототипу.

Запропоновано функціональну модель модульної IoT-архітектури, що базується на розмежуванні рівнів збору, обробки, управління та візуалізації даних. Реалізовано ключові компоненти системи, включаючи алгоритм протокольної нормалізації, механізми вибору локальних контролерів на основі алгоритму DLC-CDS, а також модулі SDECR та SDESer для динамічної обробки задач та подій.

Розроблено прототип шлюзового модуля з підтримкою мультипротокольного обміну даними, локальним зберіганням, нормалізацією трафіку та передачі його до аналітичної надбудови. Запропоновані технічні рішення дозволяють інтегрувати пристрої різних виробників, підвищити адаптивність системи до змін у середовищі та зменшити складність масштабування.

Ключові слова: IoT, SDN, ШЛЮЗОВИЙ МОДУЛЬ, ПРОТОКОЛЬНА НОРМАЛІЗАЦІЯ, МОДУЛЬНА АРХІТЕКТУРА, ІНТЕЛЕКТУАЛЬНА СИСТЕМА, SDECR, SDESer, DLC-CDS.

ANNOTATION

Qualification Thesis on the Topic: "Software-driven modular IoT Architecture" submitted for the attainment of the Bachelor's Degree in the specialty 122 "Computer Science" within the educational program "Computer Science" consists of 56 pages and includes 17 illustrations, 3 appendices, and 46 references.

The aim of this qualification thesis is to develop a software-defined modular IoT architecture using Software-Defined Networking (SDN) approaches, which ensures protocol normalization, data storage, integration with sensor infrastructure, and interaction with analytics and visualization modules.

The work applies methods for analyzing architectures and algorithms for Internet of Things systems, modeling the structure of gateway modules, designing data normalization algorithms, using system programming tools, and employing visualization and prototype testing instruments.

A functional model of a modular IoT architecture is proposed, based on the separation of data acquisition, processing, management, and visualization layers. Key system components have been implemented, including a protocol normalization algorithm, mechanisms for selecting local controllers based on the DLC-CDS algorithm, and the SDECR and SDESer modules for dynamic task and event processing.

A prototype of the gateway module has been developed, supporting multiprotocol data exchange, local storage, traffic normalization, and transmission to the analytical layer. The proposed technical solutions enable the integration of devices from different manufacturers, enhance system adaptability to environmental changes, and reduce the complexity of scaling.

Keywords: IoT, SDN, GATEWAY MODULE, PROTOCOL NORMALIZATION, MODULAR ARCHITECTURE, INTELLIGENT SYSTEM, SDECR, SDESer, DLC-CDS.

ЗМІСТ

ВСТУП

«Інтернет речей (IoT)» означає глобальну мережу об’єктів або «речей», безперебійно під’єднаних до Інтернету, які можуть кардинально змінити спосіб взаємодії з навколишнім середовищем. Цей IoT дозволяє фізичним об’єктам бачити, слухати, думати та виконувати завдання шляхом обміну інформацією та координації рішень. IoT перетворює об’єкти з традиційних на розумні, використовуючи базові технології, такі як сенсорні мережі, вбудовані пристрої, комунікаційні технології, а також всюдисущі та поширені обчислення.

Зростаюча кількість фізичних об’єктів підключається до Інтернету з безпрецедентною швидкістю, що призводить до різноманітного спектру застосувань, включаючи, але не обмежуючись, розумні міста [1], розумні будинки [2], точне землеробство [3], структурний моніторинг здоров’я, дистанційне охорону здоров’я [4], розумне управління водою та енергією [5] та розумні автомобілі.

Це призвело до розробки складних і ефективних програм і подальших технологічних проблем. Додатки, розроблені на основі цих нових технологій, мають бути протестовані, перевірені та вдосконалені перед розгортанням у режимі реального часу. Інструменти моделювання корисні, оскільки пропонують швидкий, гнучкий і економічний спосіб перевірки поведінки програми. Однак вони призводять до припущень щодо кількох ключових факторів у тестовому середовищі, що призводить до величезної невизначеності. Програми IoT і бездротові сенсорні мережі дуже сильно залежать від непередбачуваних подій і фізичних характеристик, які часто призводять до неточних результатів під час моделювання.

Відповідно, існує сильний попит на розгортання систем Інтернету речей у реальному часі, проведення апаратних експериментів і використання відповідних інструментів для експериментів. Спираючись на це, багато дослідників розгорнули кілька тестових платформ із різними архітектурами, апаратним забезпеченням і топологіями для різноманітних застосувань. Оскільки IoT включає в себе

взаємозв’язок величезної кількості пристроїв від багатьох виробників і галузей, а стратегія продуктивності значно змінюється залежно від різних програм і вимог користувачів, неоднорідність пристроїв і інформації має першорядне значення. Відповідно, архітектура була основою системи IoT, і традиційна архітектура Інтернету потребує перегляду, щоб відповідати новим вимогам IoT [6]. Крім того, вкрай важливо мати гнучку багатошарову архітектуру, особливо в той час, коли постійно зростаюча кількість архітектур ще не наблизилася до еталонної моделі [7].

Численні архітектури IoT представлені в літературі [8], кожна з яких налаштована для вирішення однієї конкретної програми, але не є модульною, щоб застосовуватись для різноманітних програм. Тому є потреба в розробці програмно керованої модульної архітектури IoT, яку можна налаштувати для задоволення потреб різних додатків IoT,

Метою кваліфікаційної роботи є розробка програмно керованої модульної архітектура IoT з використанням підходів програмно-конфігурованих мереж (SDN), яка забезпечує протокольну нормалізацію, збереження даних, інтеграцію з сенсорною інфраструктурою та взаємодію з модулями аналітики і візуалізації.

Завдання кваліфікаційної роботи полягають у послідовному виконанні наступних етапів:

– проаналізувати існуючі програмно керовані архітектури;

– провести аналіз існуючих рішень;

– сформувати постановку задачі;

– розробити концепцію багаторівневої програмно керованої архітектури IoT;

– розробити функціональну модель шлюзового модуля, яка включає алгоритм протокольної нормалізації, буферизацію, мультипротокольну взаємодію та автоматичне визначення типу трафіку;

– реалізувати механізм взаємодію шлюзу з рівнями системи;

– обґрунтувати вибір алгоритмів для вибору локального контролера в динамічних мережах;

– реалізувати мінімальну програмну реалізацію частини шлюзового модуля з протокольною нормалізацією;

– провести тестування шлюзового модуля з протокольною нормалізацією.

Об’єкт дослідження - багаторівнева інформаційна система Інтернету речей (IoT), що включає сенсори, шлюзові пристрої, хмарну або серверну інфраструктуру для обробки і візуалізації даних, а також засоби керування мережею.

Предмет дослідження - алгоритми, архітектурні рішення та програмні модулі, які реалізують функціональність шлюзового модуля в IoT-системі з підтримкою SDN-технологій.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах студентської науково-практичної конференції “Інтелектуальні інформаційні технології в прикладних дослідженнях” (IITAR – 2025), м. Тернопіль, Україна, 27-29 травня 2025 р.

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

1 АНАЛІЗ ПРОГРАМНО КЕРОВАНИХ АРХІТЕКТУР IOT

1.1 Опис програмно керованих архітектур

Сучасні тенденції розвитку Інтернету речей (IoT) орієнтовані на побудову гнучких, масштабованих, адаптивних систем, що здатні інтегрувати гетерогенні пристрої, сенсори та протоколи в єдину функціональну інфраструктуру. У такому контексті особливого значення набувають програмно керовані архітектури, що дозволяють централізовано або розподілено контролювати та конфігурувати компоненти IoT-середовища.

Потреба в таких рішеннях викликана високим ступенем гетерогенності IoT-систем, що складаються з різнорідних пристроїв, які працюють за різними протоколами, мають різний рівень обчислювальної потужності та енергоспоживання. Програмно керовані підходи дозволяють абстрагувати апаратну частину та забезпечити централізоване управління шляхом використання відповідних інтерфейсів прикладного програмування (API) і політик.

Розвиток IoT привів до збільшення кількості взаємопов’язаних пристроїв, які збирають та обмінюються величезними обсягами даних. Ці пристрої потребують ефективного управління, обробки даних у реальному часі та адаптації до зміни умов середовища. У зв’язку з цим виникла концепція програмно визначених архітектур, зокрема програмно визначених мереж (Software-Defined Networking, SDN), яка дозволяє динамічно керувати ресурсами та функціональністю IoT-систем, забезпечуючи при цьому гнучкість, масштабованість і безпеку [9].

В роботі [10] представлено архітектуру розумного шлюзу для індустріального IoT, що базується на концепції програмного визначення. Такий шлюз дозволяє адаптивно конфігурувати параметри інтерфейсів, протоколів збору та передачі даних, а також підтримує модульну розробку програмного забезпечення. Основними рівнями архітектури є: апаратна частина з ARM-процесором, базове програмне забезпечення на основі Linux та модульна прикладна система, що включає драйвери протоколів, внутрішню базу даних (MemDB), модуль керування драйверами та підтримку протоколів OPC UA і

MQTT. Завдяки конфігураційним XML-файлам забезпечується швидка інтеграція нових пристроїв.

Автори в досліджені [11] розробили платформу MINOS як приклад багатопротокольного SDN-рішення для IoT. Платформа підтримує централізоване логічне управління мережею та дозволяє динамічно конфігурувати протоколи на окремих вузлах. Використано власні реалізації протоколів CORAL-SDN і Adaptable-RPL, що підвищують доставку пакетів при зниженні накладних витрат. Важливо, що система має графічний інтерфейс користувача для візуалізації та керування топологіями IoT-мереж. Зазначено, що гнучкість SDN дозволяє ефективно управляти ресурсами в умовах великої кількості IoT-вузлів, зменшуючи складність конфігурації та покращуючи продуктивність системи.

В роботі [12] автори розглядали концепцію "software-defined IoT units", що абстрагують доступ до IoT-ресурсів і забезпечують динамічне налаштування функціональності. Пропонується створення єдиного уніфікованого API для роботи з віртуалізованими IoT-ресурсами у хмарному середовищі. Це дозволяє масштабувати системи, забезпечити самообслуговування та орієнтацію на політики користувача. У статті підкреслюється, що в умовах динамічних змін вимог до інфраструктури, саме програмно визначене керування є ключовим чинником успішного розгортання та обслуговування IoT-сервісів.

Розробники в своєму досліджені [13] акцентують увагу на використанні SDN для підвищення надійності, масштабованості та адаптивності індустріальних IoT-мереж. У статті аналізуються підходи до централізованого та розподіленого контролю, особливості вибору SDN-контролерів та формування архітектур з урахуванням вимог IIoT (Industrial Internet of Things). Авторами представлено огляд існуючих архітектур SDN для IIoT, а також класифікацію контролерів відповідно до їх функціональних можливостей і сумісності з різними типами мережевих топологій. У роботі також порушуються питання забезпечення QoS, безпеки та балансування навантаження в системах, що динамічно змінюються.

Загалом, предметна область програмно керованої модульної IoT-архітектури охоплює ряд важливих аспектів, зокрема:

- використання абстракції апаратного рівня через API, що дозволяє приховати складність взаємодії з різнорідними пристроями та забезпечити стандартизований підхід до розробки додатків;

- розділення функціональних рівнів (апаратний, базове ПЗ, прикладний рівень) задля досягнення гнучкості та повторного використання компонентів;

- підтримку мультипротокольної взаємодії (OPC UA, MQTT, CoAP, CORAL-SDN, RPL) для забезпечення сумісності з різними IoT-платформами та пристроями;

- реалізацію інструментів керування та візуалізації (GUI, XML-конфігурація), що дозволяє спростити налаштування, моніторинг та усунення несправностей;

- можливість централізованого або розподіленого управління компонентами системи, залежно від потреб конкретного домену застосування;

- підтримку гнучкого розгортання та динамічного масштабування, що є критично важливим для сценаріїв з великою кількістю пристроїв та динамічними навантаженнями.

Перевагою таких архітектур є їхня здатність до адаптації у складних індустріальних середовищах, забезпечення QoS, безпеки та інтероперабельності. Програмно визначені шлюзи та IoT-одиниці можуть забезпечити швидке оновлення функціональності, підтримку нових протоколів без зміни апаратного забезпечення, що знижує витрати та спрощує супровід системи. Зокрема, використання SDN-дозволяє централізовано керувати маршрутами передачі даних, політиками доступу та безпеки, що особливо важливо для критичних застосувань на кшталт медицини, енергетики та промисловості [14, 15].

Окрім цього, гнучкість модульної архітектури дозволяє застосовувати підхід орієнтований на мікросервіси, де кожна функціональна одиниця представлена окремим модулем з чітко визначеним інтерфейсом. Це не лише покращує масштабованість та повторне використання, а й дозволяє швидко вносити зміни в архітектуру без порушення роботи всієї системи [16].

Таким чином, формування програмно керованих IoT-систем є ключовим напрямом у розробці інфраструктур розумного виробництва, розумних міст, адаптивної охорони здоров’я, енергетики, логістики, агропромисловості та інших

доменів цифрової трансформації. Застосування SDN, віртуалізації, мікросервісної архітектури та стандартизованих протоколів дозволяє створювати системи нового покоління, які здатні самостійно адаптуватися до змін у середовищі, реагувати на події в режимі реального часу та задовольняти вимоги користувачів з високою точністю та надійністю.

1.2 Огляд і аналіз існуючих рішень

Перша категорія представляє пов’язані архітектури IoT, які були запропоновані для охорони здоров’я та медичних додатків. Використання Інтернету речей в охороні здоров’я набуло першочергового значення в останні роки, спрямоване на забезпечення профілактичної медицини та збереження добробуту. IoT надає технологічні рішення для побудови мереж поінформованих і пов’язаних електронних пацієнтів, за допомогою яких спілкування між пацієнтами та постачальниками медичних і соціальних послуг може відбуватися в режимі реального часу [17]. Основною метою таких платформ є надання недорогих, малопотужних, надійних і переносних пристроїв, які збирають необхідну інформацію для медичного, активного життя та допомоги.

Дослідження в [18] представляє WSN на основі Інтернету речей для моніторингу тепличного середовища. Автори стверджують, що фреймворк також можна масштабувати для інших додатків моніторингу, а крайові вузли мають можливість підключати додаткові датчики. Крім того, необроблені значення RSSI використовуються для визначення наближення, головними обмеженнями є послідовність і точність. Крім того, інші програми не тестувалися на цій платформі. Вирішення цих проблем вимагає обширних математичних алгоритмів [19], які неможливо реалізувати на запропонованих архітектурах.

Точне землеробство (PA) — ще одна сфера застосування, де IoT підвищує ефективність, продуктивність і прибутковість багатьох систем сільськогосподарського виробництва. Інформація про навколишнє середовище в режимі реального часу збирається дистанційно з сільськогосподарських середовищ

і передається туди, де її можна обробити для виявлення проблем, збереження даних або виконання необхідних дій. Відповідно, було розроблено декілька систем IoT для моніторингу точного землеробства [20]. Платформи IoT нещодавно були розроблені для контролю споживання води при зрошенні. Розумна зрошувальна система проілюстрована в [21], де автори розробили та впровадили хмарну систему моніторингу поля, яка інформуватиме фермера про стан ґрунту на полі та була впроваджена на фермі.

Одним із основних додатків, які можуть використовувати IoT, є локалізація цілей у приміщенні та відстеження об'єктів. Системи локалізації та відстеження цілей відіграють вирішальну роль у кількох контекстно-орієнтованих програмах, надаючи важливу інформацію для позиціонування, відстеження та навігації, де глобальна система позиціонування (GPS) зазвичай неможлива в приміщенні через поганий супутниковий прийом. Це призвело до розробки безлічі практичних застосувань, таких як локалізація всередині приміщень у розумних будівлях [22], навігаційні системи для сліпих [23] та автономні навігаційні роботи [24].

У [25] автори розробили та реалізували систему збору даних для IoT. Система моделюється як прототип платформи моніторингу, але аналіз ефективності не проводиться. Структура на основі IoT представлена в [26] для полегшення навчання локалізації в приміщеннях із застосуванням бездротових модулів Arduino та XBee для виконання локалізації в приміщенні за допомогою тріангуляції на основі RSSI. Це використовується для визначення положення та динамічної карти вузлів у приміщенні, але має обмежену точність і ефективність.

Вищезазначені роботи зосередили свою роботу на деяких конкретних функціях або конкретних програмах IoT, але не зосереджувалися на тестуванні своєї платформи в інших програмах, обмежуючи масштабованість і модульність архітектур.

Важливо враховувати, що будь-яка запропонована система повинна використовувати мінімальні ресурси та бути легкою для впровадження. Коліос та ін. [27] запропонував архітектуру ініціювання подій, згідно з якою система складається з віддаленого та локального хостів, а подія ініціюється на основі будь-

яких непередбачених дій, виявляється та контролюється за допомогою трьох фаз ініціювання та обробки події. Це дозволяє комунікації та обчисленню відбуватися лише тоді, коли сталася подія, дозволяючи менше використовувати ресурси та втручання людини. Модель здатна до контекстуалізації, коли модель може бути приєднана до додаткового сенсорного вузла, який може збирати контекст і дані датчика з мобільного телефону. Контекст можна зберігати в аналітиці даних і обробляти. Однак однією з основних проблем запропонованої архітектури є те, що вона не є портативною, а також не розроблена для додатків IoT з низьким енергоспоживанням. Автори в роботі [28] запропонували легко адаптовану структуру, засновану на контекстно-залежному тригері, який збирає дані зі смартфонів користувача, такі як місцезнаходження та моделі руху, і надає пропозиції, наприклад напрямки. Однак його реалізація ускладнюється можливістю розширення для включення різноманітних датчиків і високоенергоємного зв'язку на основі Wi-Fi між пристроями.

В дослідженні [29] запропоновано полегшену структуру, інтегровану з мобільним додатком, який може підключати гетерогенні пристрої. Він підтримує лише платформу Android і використовує мову розмітки датчиків для взаємодії як з приводами, так і з датчиками, а також використовує спеціальний проксі для кожного каналу зв'язку. Однак, оскільки використовується семантичне міркування, запропонована модель потребує потужності та високої обчислювальної потужності, що не завжди можливо в додатках IoT.

Метою архітектури IoT є збір і використання інформації з середовища для моніторингу, контролю, оптимізації та автоматизації програм. Однак процес виконання цих операцій часто не є простим. По-перше, фізична сутність має бути відчута перетворювачем, а зміна електричних властивостей має бути перетворена в числове значення, що виконується шаром фізичних датчиків і виконавчих механізмів. Ці необроблені дані потрібно відфільтрувати, спочатку обробити на абстрактному рівні та виконати це за допомогою низькопотужного рівня вбудованого процесора. Оскільки дані отримуються з різних віддалених місць у середовищі, бездротові приймачі використовуються для надсилання даних від

кількох вбудованих процесорів із низьким енергоспоживанням до локального інтернет-шлюзу. Цей інтернет-шлюз додатково аналізує та передає інформацію на хмарний сервер керування додатками для зберігання, детального аналізу даних і відгуків користувачів. Таким чином, різні завдання повинні виконуватися на кожному рівні архітектури.

Існує кілька архітектур, які складаються з емуляторів, симуляторів і фізичного середовища, що забезпечує безпеку та доступність. Великомасштабні проекти можна легко розгортати за допомогою розгортання на кількох сайтах або об'єднання. Тестові стенди, такі як NET Eye та IoT-LAB, є хорошими системами планування та можуть легко налаштувати вибрані вузли [30]. Крім того, ці існуючі моделі складаються з дуже небагатьох сенсорних вузлів, а створеними даними вручну керують системні адміністратори. Деякі дослідження [31] були запропоновані для вирішення цієї вимоги. У [32] автори описали трирівневу архітектуру, що включає сприйняття, мережу та прикладний рівень. Однак архітектура лише з трьома рівнями включає багато елементів, які перетинають кілька рівнів, і це робить реалізацію стандартних компонентів складним завданням. Подібним чином в [33] представили п'ятирівневу архітектуру для бізнес-додатків, а в [34] представлено чотирирівневу архітектуру, де елемент перетинає кілька рівнів, що робить реалізацію в реальному часі складним завданням. Навпаки, запропонована архітектура побудована на п'яти рівнях, де кожен рівень відповідає за кілька простих і легко визначити завдань, як показано на малюнках 1 і 2. Це забезпечує модульність на кожному рівні та робить реалізацію в реальному часі простим завданням.

В роботі [35] наголошується, що традиційні IoT-архітектури, зокрема трирівневі моделі (вузол-шлюз-хмара), мають низьку адаптивність та обмежену здатність до управління потоками даних. Автори пропонують архітектуру на базі програмно визначеної мережі (SDN), яка забезпечує розмежування контрольної та передавальної площин і дозволяє більш гнучко керувати пристроями. Така архітектура враховує особливості IoT-мереж, зокрема обмеженість ресурсів, мобільність вузлів та потребу в підтримці QoS. Їхня модель орієнтована на

використання SDN-контролерів, які розширюють можливості для адаптивного маршрутизаційного планування, обробки даних на периферії та динамічного моніторингу трафіку.

У дослідженні [36] представлено MINOS — платформу, що дозволяє здійснювати централізоване управління IoT-середовищем через SDN. Однією з ключових інновацій є підтримка кількох протоколів маршрутизації одночасно (CORAL-SDN, RPL), а також механізми моніторингу мережевого стану. Автори підкреслюють, що застосування програмно визначеної логіки дозволяє знизити затримки, підвищити ефективність передачі даних і забезпечити стабільну роботу навіть в умовах великої кількості вузлів. MINOS також передбачає модульну конфігурацію програмних компонентів, що відповідає за контроль доступу, облік трафіку, балансування навантаження та виявлення збоїв у мережі.

В дослідженні [37] запропонувано архітектуру програмно визначеного шлюзу для індустріальних IoT-рішень. Цей шлюз реалізує тришарову структуру: апаратне забезпечення (на основі ARM-процесора), базову операційну систему (Linux) і прикладні модулі (протокольні драйвери, база даних, контролер). Важливою перевагою є підтримка гнучкого конфігурування пристроїв за допомогою XML-файлів, що значно полегшує інтеграцію нових сенсорів у мережу. Крім того, шлюз підтримує MQTT і OPC UA, що дозволяє поєднувати пристрої з різними характеристиками. Це рішення також забезпечує локальну обробку даних, що знижує затримку та зменшує обсяг трафіку в мережі.

В іншій роботі [38] розглядається ідея "software-defined IoT units", які працюють як логічні представлення фізичних пристроїв. Завдяки такій абстракції можлива динамічна конфігурація пристроїв, керування політиками доступу та формування сервісів на вимогу. Пропонується створення уніфікованого API для доступу до віртуалізованих IoT-ресурсів, що забезпечує масштабованість, багаторазове використання та хмарну орієнтацію. У фреймворку SD-IoT також пропонується автоматичне виявлення ресурсів і самоконфігурація сервісів, що значно спрощує розгортання нових додатків в edge-середовищах.

В огляді [39] проведено порівняльний аналіз архітектур SDN у контексті Industrial IoT. Дослідники виділяють кілька підходів до реалізації контролерів: централізовані, ієрархічні та розподілені. Кожен з них має свої переваги: централізовані — прості у керуванні; ієрархічні — масштабовані; розподілені — більш стійкі до збоїв. У роботі аналізується використання таких платформ, як OpenDaylight, ONOS і Ryu, а також питання безпеки та QoS в умовах високої навантаженості. Автори акцентують увагу на необхідності підтримки сервісної орієнтації та мікросервісної архітектури, що дозволяє швидше реагувати на зміни запитів користувачів і навантаження системи.

Іншим важливим напрямом є застосування SDN у контексті кіберфізичних систем. У цьому випадку програмно визначені мережі дозволяють координувати обмін інформацією між сенсорами, виконавчими механізмами та аналітичними модулями в реальному часі. В роботі [40] зазначено, що це забезпечує не лише високий рівень адаптивності, але й дозволяє динамічно змінювати маршрути передачі даних, забезпечуючи стійкість системи до збоїв та атак.

Окрему увагу в роботах приділено питанням взаємодії різних протоколів передачі даних. Наприклад, в [41] демонструють, як одночасна підтримка RPL і CORAL-SDN забезпечує більшу надійність мережі. Застосування гібридних моделей дозволяє одночасно використовувати переваги SDN і традиційних IoT-протоколів. Це створює передумови для створення адаптивних систем, які автоматично перемикаються між протоколами залежно від умов середовища або вимог користувача.

Ще одним важливим аспектом є підтримка візуалізації, моніторингу та конфігурації мережі через користувацький інтерфейс. Дослідники в роботі [42] наголошується на важливості XML-конфігурації для швидкого розгортання нових пристроїв, тоді як в [43] пропонують графічний інтерфейс, що дозволяє наочно спостерігати за топологією мережі та станом вузлів. Такі рішення спрощують керування навіть у складних системах і знижують вимоги до технічної підготовки операторів. У рамках MINOS передбачено інтеграцію з системами логування та сигналізації, що дозволяє своєчасно виявляти аномалії у мережевому трафіку.

Усі ці підходи свідчать про те, що програмно керовані архітектури мають значний потенціал для IoT. Вони дозволяють ефективно управляти ресурсами в умовах гетерогенного середовища, підвищують гнучкість систем, прискорюють розгортання нових сервісів і спрощують інтеграцію пристроїв від різних виробників. Крім того, модульний підхід забезпечує високу масштабованість системи, дозволяючи створювати як локальні рішення для окремих доменів, так і глобальні розподілені інфраструктури.

Підсумовуючи аналіз літератури, можна виділити основні характеристики успішних рішень у сфері програмно керованих модульних IoT-архітектур:

- модульність на всіх рівнях (від фізичних сенсорів до сервісів);
- гнучкість в інтеграції нових компонентів;
- підтримка множинних протоколів та стандартизованих API;
- централізоване або ієрархічне управління з можливістю масштабування;
- адаптивність до навантажень і умов середовища;
- інструменти моніторингу, візуалізації та динамічного налаштування;
- хмарна орієнтованість та підтримка сервісних моделей (PaaS, IaaS);
- підтримка self-configuration та self-healing механізмів.

Застосування таких архітектур дозволяє вирішити низку ключових проблем у сфері IoT: зниження енергоспоживання, забезпечення QoS, підвищення безпеки, адаптацію до динамічних змін та зменшення вартості підтримки інфраструктури. У результаті програмно керовані IoT-системи стають потужним інструментом для реалізації концепцій розумного міста, індустрії 4.0, цифрового сільського господарства та медичних сервісів нового покоління.

1.3 Постановка задачі

Інформаційно-аналітичні системи, що базуються на IoT-технологіях, дедалі частіше використовуються в різних галузях — від сільського господарства до розумних міст і промислових підприємств. Разом із цим зростає потреба у створенні програмно керованої архітектури, яка б забезпечила масштабованість,

адаптивність, енергоефективність та високий рівень інтелектуальної обробки даних. Сучасні підходи до побудови таких систем часто мають жорстку структуру, залежність від конкретних технологій та обмеження у гнучкості налаштувань. Більшість реалізацій не передбачає динамічного розширення функціональності, а зміна конфігурації потребує ручного втручання, що ускладнює адаптацію до нових умов роботи.

У відповідь на зазначені проблеми актуальним є створення концепції програмно керованої модульної архітектури IoT-системи, яка передбачає чітке розмежування рівнів збору, обробки, управління та представлення даних. Така архітектура повинна включати інтелектуальні шлюзи з попередньою обробкою сигналів, програмно визначені мережі (SDN) для гнучкого управління трафіком, розподілене планування задач на обчислювальних вузлах та сервіси обробки подій із застосуванням rule-based inference.

Метою цієї роботи є розробка програмно керованої модульної архітектури IoT, яка реалізує модульний підхід до управління обчисленнями та взаємодії пристроїв на всіх рівнях — від фізичних сенсорів до хмарних аналітичних сервісів.

Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Розробити концепцію багаторівневої програмно керованої архітектури IoT.
2. Розробити функціональну модель шлюзового модуля, яка включає алгоритм протокольної нормалізації, буферизацію, мультипротокольну взаємодію та автоматичне визначення типу трафіку.
3. Реалізувати механізм взаємодію шлюзу з рівнями системи.
4. Обґрунтувати вибір алгоритмів для вибору локального контролера в динамічних мережах;
5. Реалізувати мінімальну програмну реалізацію частини шлюзового модуля з протокольною нормалізацією.
6. Провести тестування шлюзового модуля з протокольною нормалізацією.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ПРОГРАМНО КЕРОВАНОЇ МОДУЛЬНОЇ АРХІТЕКТУРИ ІОТ

2.1 Концепція програмно керованої модульної архітектури IoT

Концепція програмно керованої модульної архітектури для Інтернету речей (IoT) базується на сучасних підходах до побудови динамічних, масштабованих та адаптивних систем, які можуть функціонувати в умовах гетерогенних технологій, змінного середовища та високих вимог до безпеки, швидкодії й доступності. Суть цієї концепції полягає в розділенні фізичних пристроїв, обчислювальних ресурсів, логіки управління та сервісів в єдину багаторівневу архітектуру, кожен рівень якої виконує окремі функції та має свої власні алгоритми взаємодії, управління та обробки даних.

Загальна ідея програмно керованої модульної архітектури (рисунок 2.1) полягає у тому, що більшість функціональних можливостей, які раніше були закладені у апаратні компоненти, тепер реалізуються через програмні засоби. Це дає змогу централізовано або розподілено управляти як самими пристроями, так і потоками даних, типами протоколів, сервісами аналітики та іншими компонентами IoT-середовища. Рішення, які приймаються в рамках цієї архітектури, дозволяють досягти більш високої продуктивності, адаптивності до змін у мережі, гнучкого розгортання нових пристроїв або сервісів без значних зусиль з боку адміністратора або розробника.

Архітектура складається з кількох рівнів, починаючи від фізичних пристроїв, що збирають дані з навколишнього середовища, до рівня хмарних сервісів, які здійснюють глибоку аналітику, візуалізацію та керування IoT-інфраструктурою. Найнижчим рівнем є фізичний рівень, на якому функціонують сенсори, актуатори, вбудовані комп’ютери або мікроконтролери. Їхнє завдання полягає у зборі даних, зчитуванні параметрів навколишнього середовища або об'єктів, виконанні дій на основі сигналів з вищих рівнів, наприклад, відкриття клапану, запуск вентилятора або сповіщення про тривогу.

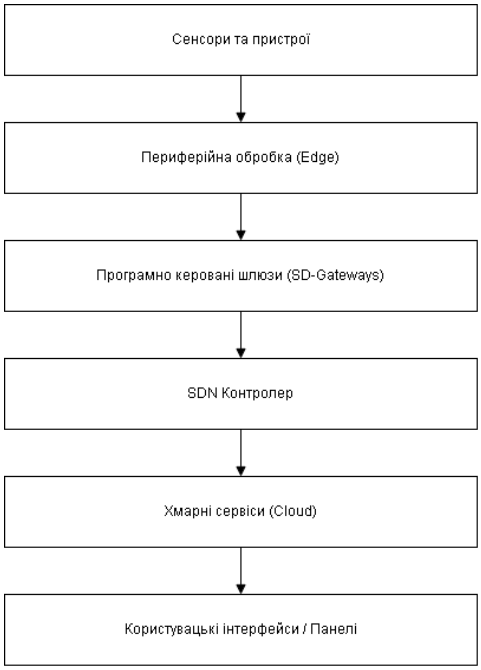


Рисунок 2.1 - Концепція програмно керованої модульної архітектури

Для ефективної роботи на цьому рівні застосовуються алгоритми енергоефективного збору даних, зокрема подієво орієнтоване зчитування (рисунок 2.2) або опитування з часовим інтервалом.

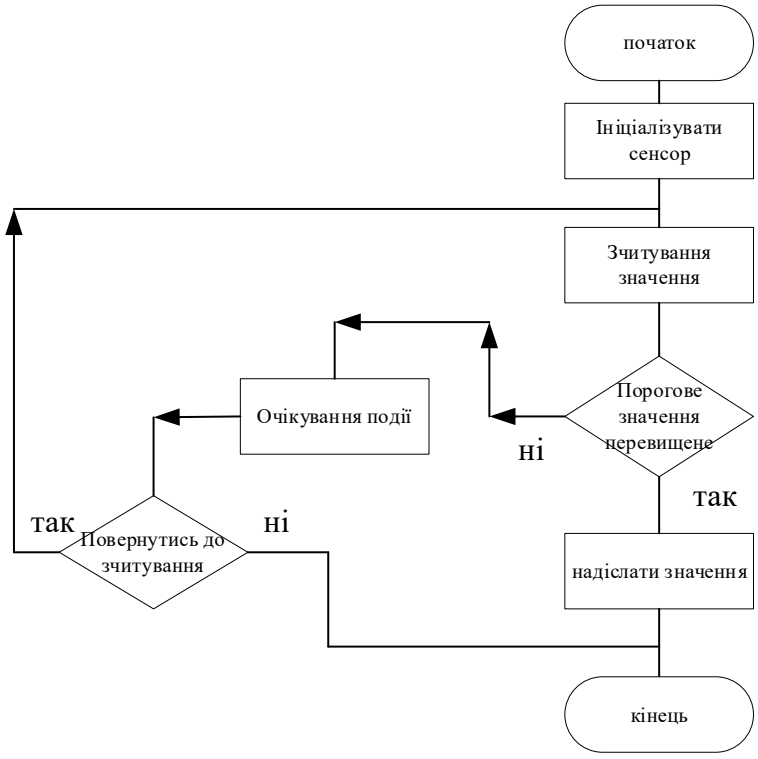


Рисунок 2.2 - Подієво орієнтоване зчитування

Деякі сенсори працюють за принципом порогу, коли передача даних відбувається лише при перевищенні встановлених значень. Це дозволяє зменшити навантаження на мережу та знизити енергоспоживання.

Дані, які зібрані фізичними пристроями, передаються до шлюзів, які розміщені на так званому рівні обробки на краю (edge level) (рисунок 2.3). У цій архітектурі кожен шлюз виконує функції попередньої обробки даних, перетворення протоколів, агрегації сигналів з кількох пристроїв та їхньої стандартизації. Для реалізації цих функцій шлюзи оснащені обчислювальними модулями, часто побудованими на базі ARM-процесорів, і програмним забезпеченням, яке дозволяє адаптуватися до різних протоколів зв'язку. Алгоритм функціонування шлюзу передбачає визначення типу вхідного протоколу (наприклад, Modbus, ZigBee, RS485), його розбір, нормалізацію структури повідомлень у єдиний формат (JSON, XML або двійковий буфер) і передачу цих повідомлень далі в систему уніфікованим способом. Крім того, реалізується буферизація даних у випадку нестабільного зв’язку з наступним рівнем. У разі використання MQTT або CoAP шлюзи можуть виконувати роль брокерів або клієнтів, що дозволяє ефективно маршрутизувати дані до відповідних служб.

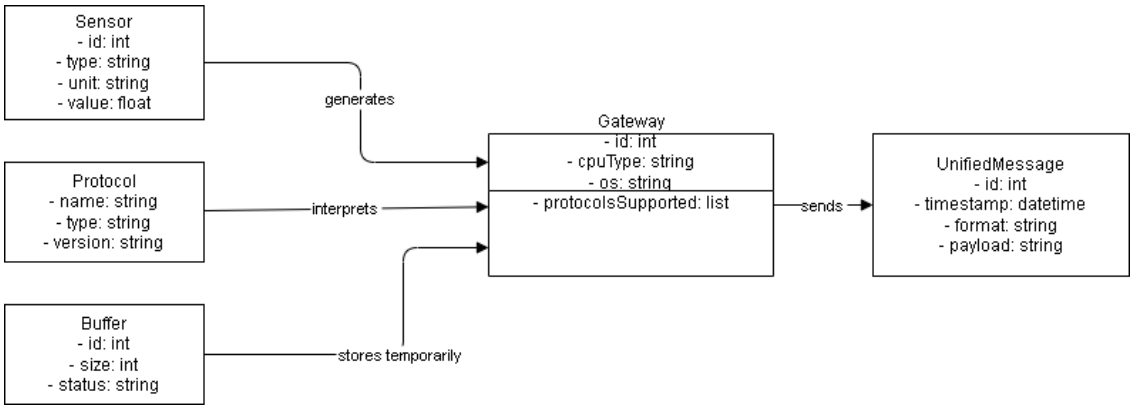


Рисунок 2.3- ER-діаграма логіки обробки даних на шлюзі в архітектурі IoT

На наступному рівні архітектури розміщується інтелектуальна система управління мережею (рисунок 2.4). Тут працюють програмно визначені мережеві контролери, які забезпечують керування маршрутизацією, пріоритезацією трафіку, балансуванням навантаження та моніторингом стану всієї інфраструктури. Для

цього використовується концепція Software-Defined Networking (SDN), що дозволяє розділити контрольну та транспортну площини. У межах цієї архітектури застосовується багаторівнева структура контролерів: головний (Principal Controller), який має глобальне бачення мережі, вторинні (Secondary Controllers), які відповідають за окремі підмережі або технології, та локальні (Local Controllers), які безпосередньо керують зв'язком із фізичними вузлами.

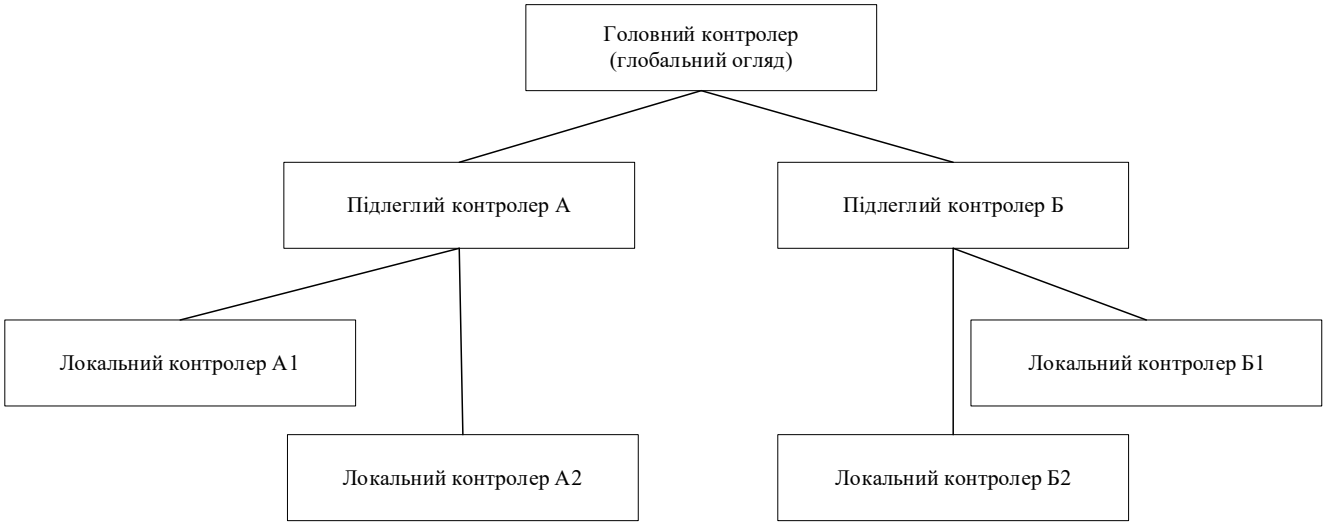


Рисунок 2.4 - Інтелектуальна система управління мережею

Вибір локальних контролерів здійснюється за допомогою алгоритму DLC-CDS (Distributed Local Controller with Connected Dominating Set) (рисунок 2.5).

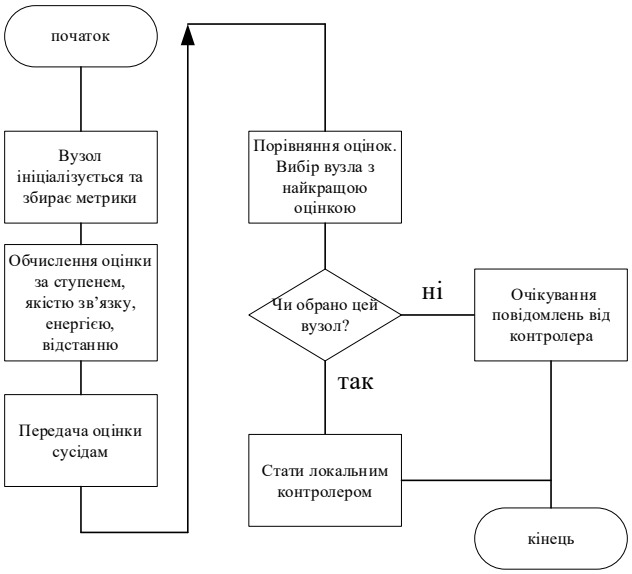


Рисунок 2.5 - Схема алгоритму вибору локальних контролерів

Цей алгоритм базується на теорії графів і застосовує нечітку логіку для оцінки кандидатів на роль контролера. Основними параметрами, які враховуються, є ступінь з'єднаності вузла, якість каналу зв’язку, рівень енергії та відстань до найближчого шлюзу. Алгоритм працює у розподіленому режимі, де кожен вузол обчислює свій власний рейтинг та передає його сусідам, після чого відбувається обрання домінантного вузла, який і стає локальним контролером. Така модель дозволяє створити ефективну систему управління навіть у умовах часткової мобільності вузлів або динамічної топології.

У межах цієї архітектури також функціонують програмно визначені обчислювальні ресурси та сервіси, які відповідають за обробку даних, виконання інтелектуальних операцій (наприклад, машинне навчання, класифікація або прогнозування) та адаптацію системи до зовнішніх впливів. Ці модулі називаються відповідно SDECR (Software-Defined Edge Computing Resource) та SDESer (Software-Defined Edge Services).

Для SDECR застосовується розподілений планувальник задач, який враховує обчислювальні можливості вузлів, поточне навантаження та пріоритетність завдань (рисунок 2.6).

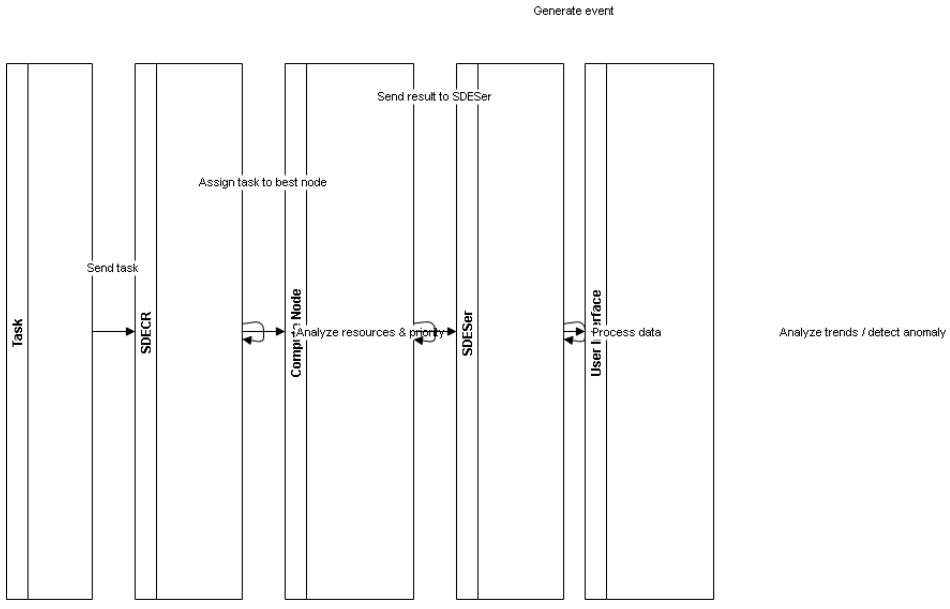


Рисунок 2.6 - Діаграма послідовності роботи модулів SDECR і SDESer у процесі обробки задач

При надходженні нової задачі, вона спочатку аналізується на предмет необхідного обсягу ресурсів, потім порівнюється з доступними вузлами в пулі, і після прийняття рішення виконується передача даних та запуск обробки. Важливим елементом тут є механізм балансування навантаження, що мінімізує затримки та перевантаження.

У свою чергу, SDESer реалізує сервіси обробки подій, наприклад, виявлення аномалій у даних, аналіз трендів, інтерпретацію сигналів на основі правил або моделей. Алгоритми в цьому блоці можуть ґрунтуватися на методах rule-based inference, fuzzy logic, або машинного навчання. Наприклад, для виявлення витоку в трубопроводі система порівнює зміну тиску з попередньо навченою моделлю нормальної поведінки, та при виявленні відхилень генерує подію та передає її на рівень сервісу або інтерфейс користувача.

Над усіма рівнями архітектури функціонує хмарна платформа або додаткова надбудова, яка виконує задачі довгострокового зберігання даних, візуалізації, налаштування правил взаємодії, а також надає API для інтеграції із зовнішніми системами, наприклад, ERP, SCADA або мобільними додатками. Усі зібрані дані передаються у центральну систему за допомогою протоколів MQTT, HTTP(S), OPC UA або WebSockets. Тут застосовуються алгоритми верифікації даних, зберігання у базах даних, генерації сповіщень та побудови дашбордів. Адміністратор може змінювати політики взаємодії, налаштовувати порогові значення, управляти пристроями, змінювати маршрутизацію або адаптувати роботу сервісів відповідно до змін у бізнес-процесах.

Таким чином, програмно керована модульна архітектура для IoT забезпечує повну адаптивність, масштабованість та сумісність у складних гетерогенних середовищах. Її перевага полягає у тому, що будь-яка складова системи може бути замінена, оновлена або масштабована без необхідності переробки всієї інфраструктури. Алгоритми, які закладено в основу функціонування модулів, забезпечують інтелектуальне управління, автономність прийняття рішень, гнучке реагування на зміну умов роботи, а також високу ефективність у плані енергоспоживання та використання мережевих ресурсів. Завдяки такому підходу,

IoT-системи нового покоління здатні самостійно адаптуватися, розширюватися та взаємодіяти з іншими цифровими екосистемами, сприяючи цифровій трансформації різних галузей — від промисловості до сільського господарства, охорони здоров'я, міської інфраструктури та інших сфер.

2.2 Функціональна модель шлюзового модуля з алгоритмом протокольної нормалізації

Шлюзовий модуль є ключовим елементом у багаторівневій IoT-архітектурі. Його головне призначення забезпечення зв'язку між сенсорним рівнем, де дані збираються з фізичного середовища, та рівнем обробки, де ці дані аналізуються. Шлюз отримує дані від сенсорів, які можуть використовувати різні протоколи зв'язку, Modbus, ZigBee або MQTT. Через цю різноманітність виникає необхідність у протокольній нормалізації — перетворенні даних у єдиний формат, зручний для подальшої обробки.

Також основним завданням шлюзу є агрегація, тобто об'єднання даних з кількох джерел для зменшення навантаження на мережу (рисунок 2.7).

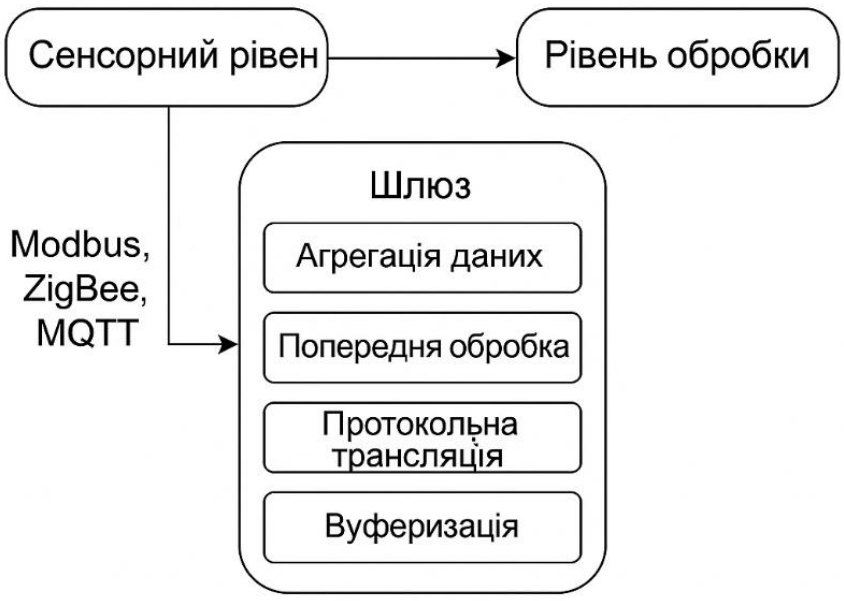


Рисунок 2.7 - Призначення та функції шлюзового модуля в IoT-архітектурі

Крім того шлюз виконує попередню обробку даних: відсів шумів, перевірку повноти та коректності. Якщо зв’язок із хмарним або серверним рівнем нестабільний, шлюз зберігає дані локально (буферизація) і передає їх при відновленні з’єднання. Тобто шлюз виступає як адаптивний посередник, який уніфікує комунікацію, підвищує ефективність передачі даних і знижує ризик втрат.

Також шлюзовий модуль складається з апаратної та програмної частин (рисунок 2.8). Апаратна частина побудована на базі ARM-процесора або системи на кристалі (SoC), що забезпечує енергоефективну та стабільну роботу.

Програмна частина шлюзу реалізує функції прийому, обробки та передачі даних. До її складу входять драйвери протоколів, які дозволяють взаємодіяти з різними типами сенсорів. Дані, що надходять, проходять через модуль протокольної нормалізації, який перетворює їх у стандартизований формат (JSON або XML). Отримана інформація зберігається у локальній базі даних, для тимчасового зберігання або буферизації. Комунікаційний стек забезпечує передачу даних до хмарних сервісів або обробних модулів за допомогою протоколів MQTT, HTTP та OPC UA.

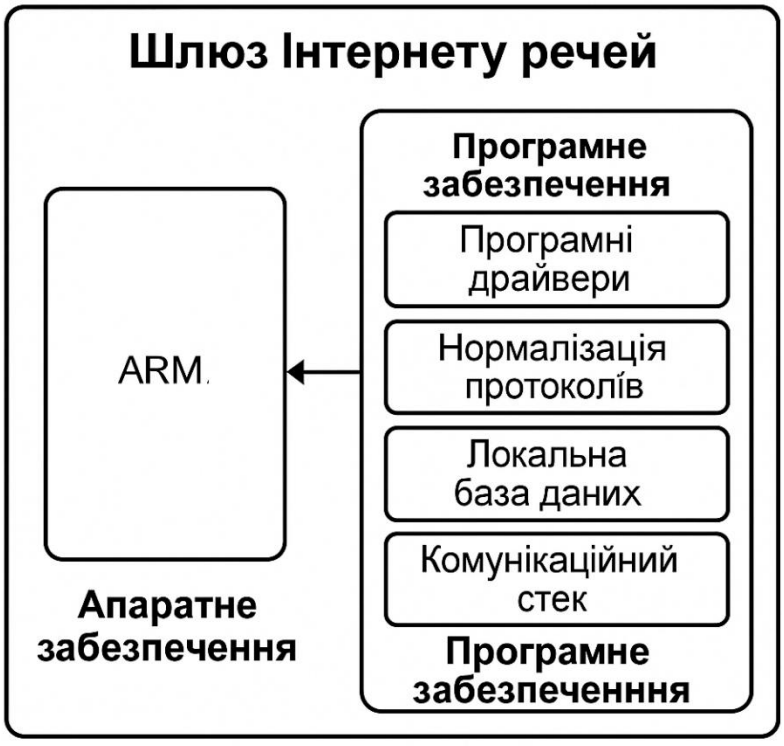


Рисунок 2.8 - Структура та основні компоненти шлюзового модуля IoT-системи

Крім того модуль має підтримувати різні комунікаційні протоколи, оскільки пристрої у системі можуть використовувати різні (рисунок 2.9).

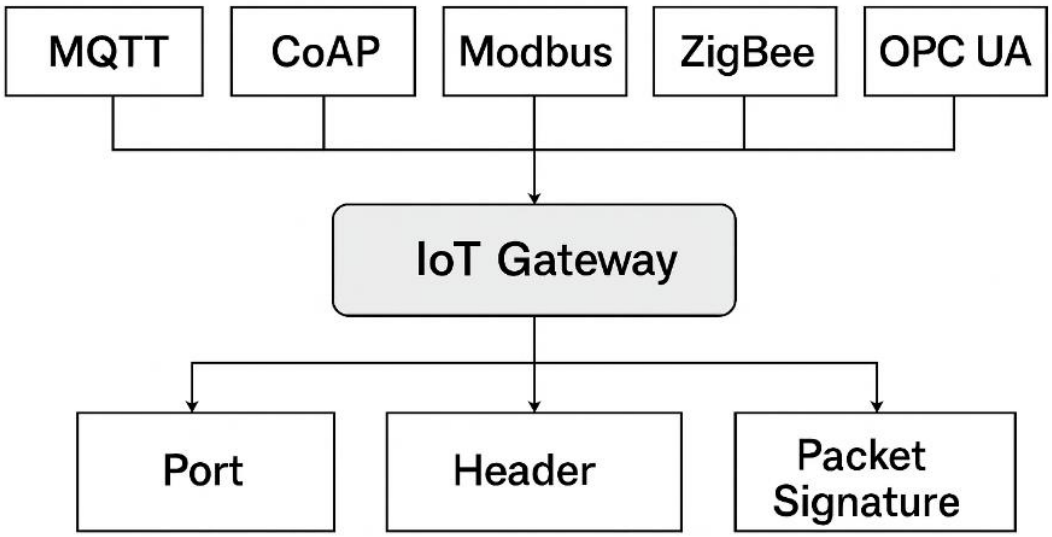


Рисунок 2.9 - Підтримувані протоколи та механізм виявлення типу вхідного трафіку в IoT-шлюзі

Найпоширенішими протоколами є MQTT, CoAP, Modbus, ZigBee, OPC UA та HTTP. Кожен із цих протоколів має свої характеристики, формат повідомлень, спосіб обміну та призначення. Для правильного оброблення вхідного трафіку шлюз повинен автоматично визначати, який протокол використовується в кожному конкретному випадку. Це завдання виконується через аналіз порту, заголовка повідомлення або сигнатури пакета.

Такий автоматичний механізм виявлення дозволяє шлюзу приймати повідомлення від різних типів пристроїв без ручного налаштування. Це значно підвищує гнучкість, масштабованість та адаптивність IoT-системи.

Алгоритм протокольної нормалізації у шлюзовому модулі призначений для перетворення вхідних повідомлень у стандартизований внутрішній формат, який зручно обробляти на наступних рівнях (рисунок 2.10).

Рисунок 2.10 - Схема алгоритму протокольної нормалізації в шлюзі IoT-системи

Вхідними даними є повідомлення, отримані з різних пристроїв, які використовують різні протоколи. Кожен протокол має власну структуру пакета, набір полів та формат представлення.

Першим етапом алгоритму є розбір структури пакета (див.рисунок 2.10).

Шлюз ідентифікує протокол та розділяє повідомлення на його складові частини. Далі здійснюється визначення ключових полів, таких як payload (власне дані), timestamp (час передачі) та device ID (ідентифікатор пристрою). Після цього виконується перетворення даних у внутрішній уніфікований формат. Найчастіше використовується JSON або XML, оскільки ці формати легко читаються та інтегруються з аналітичними сервісами.

Результатом алгоритму є стандартизоване повідомлення, яке відправляється на обробку або зберігання. Це дозволяє об'єднати дані від різних пристроїв у єдиний інформаційний потік, зменшуючи складність подальшої обробки.

2.3 Взаємодія шлюзу з рівнями системи

Шлюзовий модуль (рисунок 2.11) у IoT-системі виконує функцію посередника між сенсорним рівнем і рівнями керування, аналітики та візуалізації.

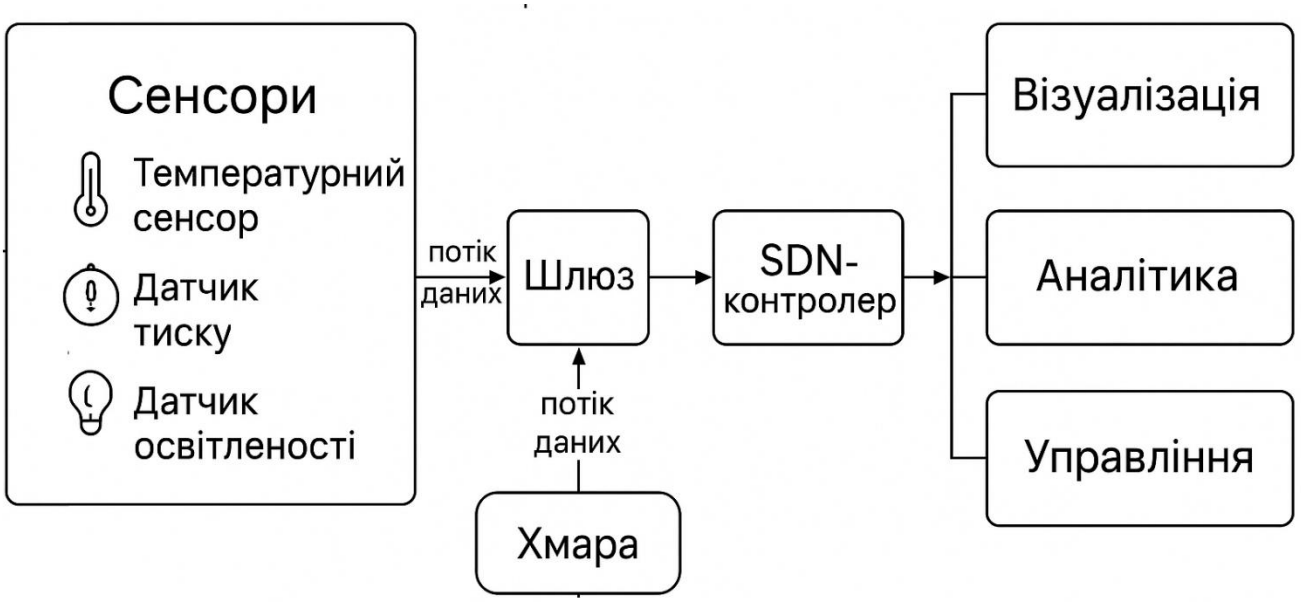


Рисунок 2.11 – Функціональна схема протокольної нормалізації в шлюзі IoT-системи

Потік даних починається із сенсорів, які вимірюють фізичні параметри середовища, температуру, вологість або тиск. Ці дані передаються до шлюзу через протоколи зв’язку, як-от ZigBee, LoRa або Modbus. У шлюзі дані проходять нормалізацію, буферизацію та попередню обробку.

Після цього оброблені повідомлення надсилаються до SDN-контролера. Контролер виконує маршрутизацію, пріоритезацію трафіку та моніторинг стану мережі. Це дозволяє динамічно керувати потоками даних у мережі з урахуванням навантаження та доступності каналів.

На наступному етапі дані передаються до хмарного середовища, де вони зберігаються, аналізуються та візуалізуються. Системи візуалізації дозволяють оператору або користувачу переглядати інформацію через дашборди. Сервіси аналітики виконують виявлення аномалій, прогнозування подій або оптимізацію процесів. Рівень управління, в свою чергу, може змінювати параметри роботи пристроїв, формувати команди управління або налаштовувати політики безпеки.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Обґрунтувати алгоритмів для вибору локального контролера в динамічних мережах

IoT-мережі часто мають динамічну топологію, яка змінюється в реальному часі. Це може бути пов’язано з мобільністю пристроїв, періодичним зникненням вузлів, відмовами обладнання або змінами в конфігурації середовища. Наприклад, у розумних містах чи в сільському господарстві пристрої можуть рухатися або тимчасово виходити з мережі. Така динамічність ускладнює централізоване управління, оскільки потребує швидкого прийняття рішень на місці, без затримок, спричинених передаванням даних до центрального сервера.

У зв’язку з цим виникає потреба у виборі локального контролера — вузла, який може тимчасово брати на себе керування сегментом мережі. Цей контролер виконує функції маршрутизації, моніторингу та координації пристроїв. Локальні рішення зменшують час реагування, покращують надійність мережі та дозволяють працювати автономно у випадку втрати зв’язку з центральною системою. Наявність механізму вибору оптимального локального контролера є критичним для стабільності та адаптивності сучасних IoT-мереж.

Для ефективного керування динамічними IoT-мережами важливо правильно обрати локальний контролер. Першим критерієм є мінімізація затримки під час прийняття рішень. Локальний контролер повинен швидко реагувати на зміни в мережі, забезпечуючи оперативну маршрутизацію та обробку подій. Чим нижча затримка, тим ефективніше працює система в режимі реального часу.

Другий важливий критерій — максимальне охоплення підмережі. Контролер має мати можливість напряму або через мінімальну кількість проміжних вузлів взаємодіяти з більшістю пристроїв у своєму кластері. Це забезпечує ефективну координацію та зменшує навантаження на мережу.

Також враховується надійність каналу зв’язку між контролером і іншими вузлами. Перевагу отримують вузли з кращим рівнем сигналу, стабільним підключенням та низькою ймовірністю відмов.

Останній критерій — енергоефективність вузла. У багатьох випадках IoT-пристрої працюють від батареї, тому контролер повинен мати достатній рівень заряду або бути підключеним до постійного джерела живлення.

Алгоритм DLC-CDS (Dynamic Load-aware Connected Dominating Set) є доцільним вибором для використання у динамічних IoT-мережах, там, де пристрої мають обмежені обчислювальні ресурси та енергоживлення. Цей алгоритм дозволяє ефективно обирати локальних контролерів з врахуванням навантаження, топології та наявних зв’язків між вузлами. Він автоматично адаптується до змін у мережі, таких як додавання нових пристроїв або вихід з ладу окремих елементів.

Порівняльний аналіз показав (таблиця 3.1), що DLC-CDS демонструє високу стійкість до збоїв, мінімальні затримки в управлінні, оптимальне охоплення підмережі та енергоефективність.

Таблиця 3.1 - Порівняння алгоритмів для вибору локального контролера

Алгоритм	Затримка прийняття рішень	Охоплення підмережі	Стійкість до збоїв	Адаптивність	Складність реалізації
Централізоване керування	Висока	Високе	Низька	Низька	Низька
Election-based (LEACH)	Середня	Середнє	Середня	Середня	Середня
MPR (Multipoint Relay)	Низька	Високе	Низька	Низька	Середня
CDS	Середня	Високе	Середня	Середня	Висока
DLC-CDS	Низька	Високе	Висока	Висока	Висока

Його характеристики повністю відповідають технічним вимогам запропонованої архітектури системи. Саме тому його вибір є обґрунтованим як для тестових прототипів, так і для подальшого масштабування.

3.2 Мінімальна програмна реалізація частини шлюзового модуля з протокольною нормалізацією

Мінімальна структура реалізації:

– sensor_receiver.py – приймає дані через різні протоколи.
– normalizer.py – уніфікує повідомлення.
– storage.py – зберігає нормалізовані дані.
– main.py – точка входу.

Модуль normalizer.py виконує функцію уніфікації даних, які надходять від різних пристроїв з використанням різних протоколів (рисунок 3.1).

```
import sqlite3

def init_db():
    conn = sqlite3.connect('gateway_data.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS data
                 (timestamp TEXT, device_id TEXT, protocol TEXT, payload TEXT)''')
    conn.commit()
    conn.close()

def store_data(json_data: str):
    import json
    data = json.loads(json_data)
    conn = sqlite3.connect('gateway_data.db')
    c = conn.cursor()
    c.execute("INSERT INTO data VALUES (?, ?, ?, ?)",
              (data['timestamp'], data['device_id'], data['protocol'], json.dumps(data['payload'])))
    conn.commit()
    conn.close()
```

Рисунок 3.1 – Модуль приведення до єдиного внутрішнього формату

Основне завдання цього модуля привести всі вхідні повідомлення до єдиного внутрішнього формату, зручного для подальшої обробки або зберігання. Функція normalize_message приймає три параметри: назву протоколу, необроблені дані у вигляді словника (dict) та ідентифікатор пристрою, який надіслав ці дані.

Після виклику функції створюється новий словник normalized, який містить чотири основні поля. Поле timestamp автоматично фіксує поточну дату і час, що дозволяє точно відстежити момент отримання даних. Поле device_id містить унікальний ідентифікатор джерела інформації. Поле protocol вказує, який саме протокол використовується для передачі даних (наприклад, Modbus або MQTT). Поле payload містить самі сенсорні дані без змін.

Отриманий словник перетворюється у формат JSON за допомогою стандартної бібліотеки json і повертається як рядок. Така нормалізація дозволяє об’єднувати дані з різнорідних джерел і готувати їх до зберігання у базі даних або передавання на інші рівні IoT-системи.

Модуль storage.py відповідає за зберігання нормалізованих даних у локальній базі даних SQLite. Він містить дві основні функції: init_db() та store_data(json_data: str) (рисунок 3.2).

```
import sqlite3

def init_db():
    conn = sqlite3.connect('gateway_data.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS data
                 (timestamp TEXT, device_id TEXT, protocol TEXT, payload TEXT)''')
    conn.commit()
    conn.close()

def store_data(json_data: str):
    import json
    data = json.loads(json_data)
    conn = sqlite3.connect('gateway_data.db')
    c = conn.cursor()
    c.execute("INSERT INTO data VALUES (?, ?, ?, ?)",
              (data['timestamp'], data['device_id'], data['protocol'], json.dumps(data['payload'])))
    conn.commit()
    conn.close()
```

Рисунок 3.2 – Модуль зберігання нормалізованих даних

Функція init_db() створює або відкриває файл бази даних з назвою gateway_data.db. Далі вона ініціалізує таблицю data, у якій передбачено чотири текстові поля: час отримання (timestamp), ідентифікатор пристрою (device_id), назву протоколу (protocol) та вміст повідомлення (payload). Якщо таблиця вже існує, вона не створюється повторно. Після цього база закривається.

Функція store_data(json_data) приймає нормалізоване повідомлення у форматі JSON. Вона декодує цей рядок у словник і вставляє значення у відповідні поля таблиці. Перед вставкою payload ще раз серіалізується як JSON, щоб зберегти структуру даних. Це дозволяє зберігати інформацію з різних сенсорів у стандартизованому форматі. Після завершення операції база даних закривається. Модуль забезпечує локальне збереження IoT-даних для подальшого аналізу або передавання.

Файл main.py виконує роль тестового скрипта для демонстрації роботи всієї системи — від отримання сирих IoT-даних до їх збереження у базі даних (рисунок 3.3).

```python
from normalizer import normalize_message
from storage import init_db, store_data

# Ініціалізація бази даних
init_db()

# Тестові "отримані" дані з пристрою по Modbus
raw_data = {"temperature": 24.5, "humidity": 60}
device_id = "modbus_sensor_001"
protocol = "ModbusTCP"

# Нормалізація і збереження
json_msg = normalize_message(protocol, raw_data, device_id)
store_data(json_msg)

print("Data processed and stored.")
```

Рисунок 3.3 – Модуль демонстації

На початку імпортуються функції з модулів normalizer і storage. Це дозволяє використовувати раніше реалізовану логіку нормалізації повідомлень та роботи з базою даних.

Першим кроком викликається функція init_db(), яка створює файл локальної бази даних gateway_data.db і таблицю data, якщо вона ще не існує. Це необхідно для подальшого збереження інформації.

Далі створюється тестовий словник raw_data, який імітує показники температури і вологості, зібрані від сенсора, що працює за протоколом Modbus. Також задаються ідентифікатор пристрою (modbus_sensor_001) і тип протоколу (ModbusTCP).

Отримані дані передаються у функцію normalize_message(), яка формує нормалізоване повідомлення у форматі JSON — зі вказаним часом, ідентифікатором, протоколом та даними. Отриманий JSON-рядок передається у функцію store_data(), яка зберігає його у відповідні поля таблиці бази даних.

На завершення виводиться повідомлення Data processed and stored., яке підтверджує, що дані були оброблені та успішно збережені. Таким чином, main.py демонструє повний цикл обробки IoT-даних від симуляції сенсора до збереження в локальне сховище.

3.3 Тестування шлюзового модуля з протокольною нормалізацією

Для перевірки працездатності програмної частини шлюзового модуля було проведено поетапне тестування в середовищі Linux на віртуальній машині. Програмна реалізація передбачає базову обробку телеметричних повідомлень з попередньою нормалізацією та збереженням у локальну базу даних. Тестування виконувалося без графічного інтерфейсу користувача, що дозволило сфокусуватися на коректності логіки обробки даних. Нижче наведено послідовність основних кроків тестування.

Основний тестовий скрипт main.py було запущено за допомогою python3 main.py (рисунок 3.4).

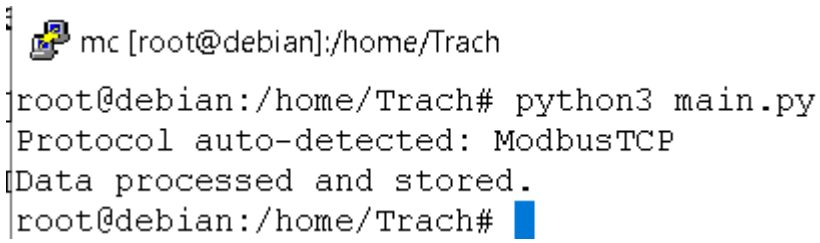


Рисунок 3.4 – Основний тестовий скрипт

Він ініціалізує базу даних SQLite, імітує отримання сирих даних з пристрою, виконує автоматичне визначення протоколу на основі структури повідомлення, нормалізує його та зберігає у базу.

В результаті успішного виконання скрипта на екран виводиться повідомлення про успішне виявлення протоколу (Protocol auto-detected: ModbusTCP) та повідомлення про збереження даних (Data processed and stored.). Це підтверджує коректну роботу основного функціоналу.

Для перевірки факту збереження інформації було відкрито SQLite CLI (рисунок 3.5) (sqlite3 gateway_data.db) та виконано запит SELECT * FROM data;. В результаті було отримано запис, що містить дату, ідентифікатор пристрою, тип протоколу та JSON-представлення даних.

```
Data processed and stored.
root@debian:/home/Trach# ls -lh gateway_data.db
-rw-r--r-- 1 root root 8,0K тра 29 22:14 gateway_data.db
root@debian:/home/Trach# sqlite3 gateway_data.db
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite> .headers on
sqlite> .mode column
sqlite> SELECT * FROM data;
timestamp                  device_id        protocol   payload

-------------------------  ---------------  ---------  -------------------------
-------------
2025-05-29T21:38:30.525124  sensor_auto_001  ModbusTCP  {"temperature": 23.1, "h
umidity": 45}
2025-05-29T21:45:54.090709  sensor_auto_001  ModbusTCP  {"temperature": 23.1, "h
umidity": 45}
2025-05-29T22:06:32.613089  sensor_auto_001  ModbusTCP  {"temperature": 23.1, "h
umidity": 45}
2025-05-29T22:08:26.795450  sensor_auto_001  ModbusTCP  {"temperature": 23.1, "h
umidity": 45}
2025-05-29T22:14:24.483600  sensor_auto_001  ModbusTCP  {"temperature": 23.1, "h
umidity": 45}
sqlite>
```

Рисунок 3.5 – Перевірка збереження в БД

Отримане повідомлення відповідало очікуваній структурі: включало ключові поля — timestamp, device_id, protocol, payload. Це свідчить про успішну роботу алгоритму нормалізації та відповідність даних внутрішньому стандарту шлюзу (рисунок 3.6).

```
mc [root@debian]:/home/Trach
+----------------------------+-----------------+------------+--------------------------------------+
| Timestamp                  | Device ID       | Protocol   | Payload                              |
+============================+=================+============+======================================+
| 2025-05-29T21:38:30.525124 | sensor_auto_001 | ModbusTCP  | {"temperature": 23.1, "humidity": 45} |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T21:45:54.090709 | sensor_auto_001 | ModbusTCP  | {"temperature": 23.1, "humidity": 45} |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T22:06:32.613089 | sensor_auto_001 | ModbusTCP  | {"temperature": 23.1, "humidity": 45} |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T22:08:26.795450 | sensor_auto_001 | ModbusTCP  | {"temperature": 23.1, "humidity": 45} |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T22:14:24.483600 | sensor_auto_001 | ModbusTCP  | {"temperature": 23.1, "humidity": 45} |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T22:29:27.106084 | sensor_001      | ModbusTCP  | {"temperature": 21.4, "humidity": 55} |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T22:29:27.463284 | sensor_002      | MQTT       | {"motion": true, "battery": 84}       |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T22:29:27.470201 | sensor_003      | HTTP       | {"light": 300, "noise": 40}           |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T22:29:27.477160 | sensor_004      | CoAP       | {"co2": 780, "tvoc": 140}             |
+----------------------------+-----------------+------------+--------------------------------------+
| 2025-05-29T22:29:27.484080 | sensor_005      | ZigBee     | {"soil_moisture": 25}                 |
+----------------------------+-----------------+------------+--------------------------------------+
root@debian:/home/Trach#
```

Рисунок 3.6 – Вивід результатів

Таким чином, базове тестування продемонструвало коректність логіки шлюзового модуля для обробки простих телеметричних повідомлень. Результати засвідчують готовність модуля до подальшого масштабування та розширення функціональності.

ВИСНОВКИ

В результаті виконання роботи отримані наступні результати:

1. Проведено аналіз сучасних програмно керованих архітектур IoT, SDN-рішення, мікросервісні підходи, концепції віртуалізації ресурсів і структури багаторівневих систем для адаптивного управління.

2. Досліджено існуючі архітектурні рішення в різних доменах застосування, виконано огляд рішень щодо їхньої масштабованості, адаптивності, енергоефективності та модульності.

3. Сформульовано постановку задачі, яка передбачає розробку багаторівневої архітектури з поділом на функціональні рівні та визначено вимоги до протокольної нормалізації, локального контролю та мультипротокольної взаємодії.

4. Запропоновано концепцію багаторівневої модульної архітектури, що включає рівень збору даних, обробки на краю, інтелектуального мережевого управління, обробки подій та сервісів візуалізації; описано функціонування кожного рівня з використанням сучасних програмно керованих підходів.

5. Розроблено функціональну модель шлюзового модуля, яка включає алгоритм автоматичного визначення типу вхідного трафіку, протокольну нормалізацію, мультипротокольну взаємодію та буферизацію, що забезпечує ефективну уніфікацію вхідних повідомлень.

6. Реалізовано механізм взаємодії шлюзового модуля з іншими рівнями архітектури, описано логіку передачі даних між компонентами, зокрема взаємодію з SDN-контролерами та обчислювальними модулями.

7. Обґрунтовано вибір алгоритму DLC-CDS для динамічного вибору локального контролера в умовах змінної топології мережі та описано логіку прийняття рішення.

8. Реалізовано програмну частину шлюзового модуля, що виконує протокольну нормалізацію.

9. Проведено тестування реалізованого шлюзового модуля в середовищі, що імітує мультипротокольний трафік.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Theodoridis E., Mylonas G., Chatzigiannakis I. Developing an IoT Smart City framework. Proceedings of the 4th IEEE *International Conference on Information, Intelligence, Systems and Applications* (IISA'13), 1–3 July 2013. Piraeus, Greece. IEEE, 2013. P. 1–6.

2. Datta S. K., Bonnet C., Gyrard A., Ferreira Da Costa R. P., Boudaoud K. Applying Internet of Things for personalized healthcare in smart homes. Proceedings of the 24th Wireless and Optical Communication Conference (WOCC'15), 21–23 October 2015. Taipei, Taiwan. IEEE, 2015. P. 164–169.

3. Sales N., Remedios O., Arsenio A. Wireless sensor and actuator system for smart irrigation on the cloud. Proceedings of the 2nd IEEE World Forum on Internet of Things (WF-IoT'15), 14–16 December 2015. Milan, Italy. IEEE, 2015. P. 693–698.

4. Maksimovic M., Vujović V., Perišić B. A custom Internet of Things healthcare system. *Proceedings of the 10th Iberian Conference on Information Systems and Technologies* (CISTI'15), June 2015. Aveiro, Portugal. IEEE, 2015.

5. Anjana S., Sahana M. N., Ankith S., Natarajan K., Shobha K. R., Paventhan A. An IoT based 6LoWPAN enabled experiment for water management. *Proceedings of the 9th IEEE International Conference on Advanced Networks and Telecommunications Systems* (ANTS'15), 16–19 December 2015. Kolkata, India. IEEE, 2015.

6. Al-Fuqaha A., Guizani M., Mohammadi M., Aledhari M., Ayyash M. Internet of Things: a survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials.* 2015. T. 17, № 4. P. 2347–2376.

7. Krco S., Pokric B., Carrez F. Designing IoT architecture(s): a European perspective. Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT'14), 6–8 March 2014. Seoul, South Korea. IEEE, 2014. P. 79–84.

8. Chen L., Yang K., Wang X. Robust cooperative Wi-Fi fingerprint-based indoor localization. IEEE Internet of Things Journal. 2016. T. 3, № 6. P. 1406–1417.

9. Jiang Z., Chang Y., Liu X. Design of software-defined gateway for industrial interconnection. *Journal of Industrial Information Integration*. 2020. T. 18. P. 100130.

10. Theodorou T., Violettas G., Valsamas P., Petridou S., Mamatas L. A Multi-Protocol Software-Defined Networking Solution for the Internet of Things. *IEEE Communications Magazine.* 2019. T. 57, № 10. PP. 42–48.

11. Nastic S., Sehic S., Le D.-H., Truong H.-L., Dustdar S. Provisioning Software-Defined IoT Cloud Systems. *Proceedings of the 2014 IEEE International Conference on Future Internet of Things and Cloud* (FiCloud), 27–29 August 2014. Barcelona, Spain. IEEE, 2014. P. 288–295.

12. Urrea C., Benítez D. Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things: A Review. *Sensors*. 2021. T. 21, № 19. P. 6585.

13. Xu B., Xu L. D., Cai H., Xie C., Hu J., Bu F. Ubiquitous data accessing method in IoT-based information system for emergency medical services. IEEE Transactions on Industrial Informatics. 2014. T. 10, № 2. P. 1578–1586.

14. Gomes T., Brito J., Abreu H., Gomes H., Cabral J. GreenMon: an efficient wireless sensor network monitoring solution for greenhouses. *Proceedings of the 2015 IEEE International Conference on Industrial Technology* (ICIT), 17–19 March 2015. Seville, Spain. IEEE, 2015. P. 2192–2197.

15. Zafari F., Papapanagiotou I. Enhancing iBeacon based micro-location with particle filtering. *Proceedings of the 58th IEEE Global Communications Conference* (GLOBECOM), 6–10 December 2015. San Diego, USA. IEEE, 2015.

16. Li J., Gu W., Yuan H. Research on IoT technology applied to intelligent agriculture // Proceedings of the 5th International Conference on Electrical Engineering and Automatic Control, July 2016. Vol. 367. IEEE, 2016. PP. 1217–1224.

17. Gu Y., Ren F. Energy-efficient indoor localization of smart hand-held devices using bluetooth. *IEEE Access*. 2015. T. 3. PP. 1450–1461.

18. Yelamarthi K., Haas D., Nielsen D., Mothersell S. RFID and GPS integrated navigation system for the visually impaired. Proceedings of the 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 1–4 August 2010. Seattle, USA. IEEE, 2010. P. 1149–1152.

19. Olszewski B., Fenton S., Tworek B., Liang J., Yelamarthi K. RFID positioning robot: an indoor navigation system. Proceedings of the 2013 IEEE International Conference on Electro/Information Technology (EIT), 9–11 May 2013. IEEE, 2013.

20. Al Faisal M. F., Bakar S., Rudati P. S. The development of a data acquisition system based on internet of things framework. Proceedings of the 2014 International Conference on ICT for Smart Society (ICISS), 27–28 September 2014. Bandung, Indonesia. IEEE, 2014. P. 211–216.

21. Atabekov A., He J., Bobbie P. O. Internet of things based framework to facilitate indoor localization education. Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), 10–14 June 2016. Atlanta, GA, USA. IEEE, 2016. PP. 269–274.

22. Kolios P., Panayiotou C., Ellinas G., Polycarpou M. Data driven event triggering for IoT applications. IEEE Internet of Things Journal. 2016.

23. Balan R. K., Misra A., Lee Y. LiveLabs: Building an in-situ real-time mobile experimentation testbed. Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile), 26–27 February 2014. Santa Barbara, CA, USA. ACM, 2014.

24. Tonneau A. S., Mitton N., Vandaele J. How to choose an experimentation platform for wireless sensor networks? A survey on static and mobile wireless sensor network experimentation facilities. Ad Hoc Networks. 2015. T. 30. PP. 115–127.

25. Pacheco J., Satam S., Hariri S., Grijalva C., Berkenbrock H. IoT security development framework for building trustworthy smart car services // Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI), 28–30 September 2016. Tucson, AZ, USA. IEEE, 2016. P. 237–242.

26. Abdmeziem M. R., Tandjaoui D., Romdhani I. Architecting the Internet of Things: state of the art. In: Robots and Sensor Clouds. Springer, 2016. P. 55–75.

27. Aazam M., Khan I., Alsaffar A. A., Huh E.-N. Cloud of things: integrating internet of things and cloud computing and the issues involved. Proceedings of the 11th

International Bhurban Conference on Applied Sciences and Technology (IBCAST), 14–18 January 2014. Islamabad, Pakistan. IEEE, 2014. PP. 414–419.

28. Bendouda M., та ін. Software-defined network architecture for the internet of things. Proceedings of the 2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech), 26–28 November 2018. Brussels, Belgium. IEEE, 2018. P. 1–7.

29. Jiang Z., Chang Y., Liu X. Design of software-defined gateway for industrial interconnection. Journal of Industrial Information Integration. 2020. T. 18. P. 100130.

30. Theodorou T., та ін. A Multi-Protocol Software-Defined Networking Solution for the Internet of Things. IEEE Communications Magazine. 2019. T. 57, № 10. P. 42–48.

31. Nastic S. Provisioning Software-Defined IoT Cloud Systems. Proceedings of the 2014 IEEE International Conference on Future Internet of Things and Cloud (FiCloud), 27–29 August 2014. Barcelona, Spain. IEEE, 2014. P. 288–295.

32. Urrea C., Benítez D. Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things: A Review. *Sensors.* 2021. T. 21, № 19. P. 6585.

33. Theodorou T., Violettas G., Valsamas P., Petridou S., Mamatas L. A Multi-Protocol Software-Defined Networking Solution for the Internet of Things. *IEEE Communications Magazine*. 2019. T. 57, № 10. P. 42–48.

34. Urrea C., Benítez D. Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things: A Review // Sensors. 2021. T. 21, № 19. PP. 6585.

35. Nastic S., Sehic S., Le D.-H., Truong H.-L., Dustdar S. Provisioning Software-Defined IoT Cloud Systems. Proceedings of the 2014 IEEE International Conference on Future Internet of Things and Cloud (FiCloud), 27–29 August 2014. Barcelona, Spain. IEEE, 2014. P. 288–295.

36. Jiang Z., Chang Y., Liu X. Design of software-defined gateway for industrial interconnection. Journal of Industrial Information Integration. 2020. T. 18. P. 100130.

37. Theodorou T., Violettas G., Valsamas P., Petridou S., Mamatas L. A Multi-Protocol Software-Defined Networking Solution for the Internet of Things. IEEE Communications Magazine. 2019. T. 57, № 10. P. 42–48.

38. Urrea C., Benítez D. Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things: A Review. *Sensors*. 2021. T. 21, № 19. P. 6585.

39. Theodorou T., Violettas G., Valsamas P., Petridou S., Mamatas L. A Multi-Protocol Software-Defined Networking Solution for the Internet of Things. *IEEE Communications Magazine*. 2019. T. 57, № 10. P. 42–48.

40. Jiang Z., Chang Y., Liu X. Design of software-defined gateway for industrial interconnection // Journal of Industrial Information Integration. 2020. T. 18. PP. 100130.

41. Theodorou T., Violettas G., Valsamas P., Petridou S., Mamatas L. A Multi-Protocol Software-Defined Networking Solution for the Internet of Things. IEEE Communications Magazine. 2019. T. 57, № 10. P. 42–48.

42. Sales N., Remedios O., Arsenio A. Wireless sensor and actuator system for smart irrigation on the cloud. Proceedings of the 2nd IEEE World Forum on Internet of Things (WF-IoT'15), 14–16 December 2015. Milan, Italy. IEEE, 2015. P. 693–698.

43. Datta S. K., Bonnet C., Gyrard A., Ferreira Da Costa R. P., Boudaoud K. Applying Internet of Things for personalized healthcare in smart homes // Proceedings of the 24th Wireless and Optical Communication Conference (WOCC'15), 21–23 October 2015. Taipei, Taiwan. IEEE, 2015. P. 164–169.

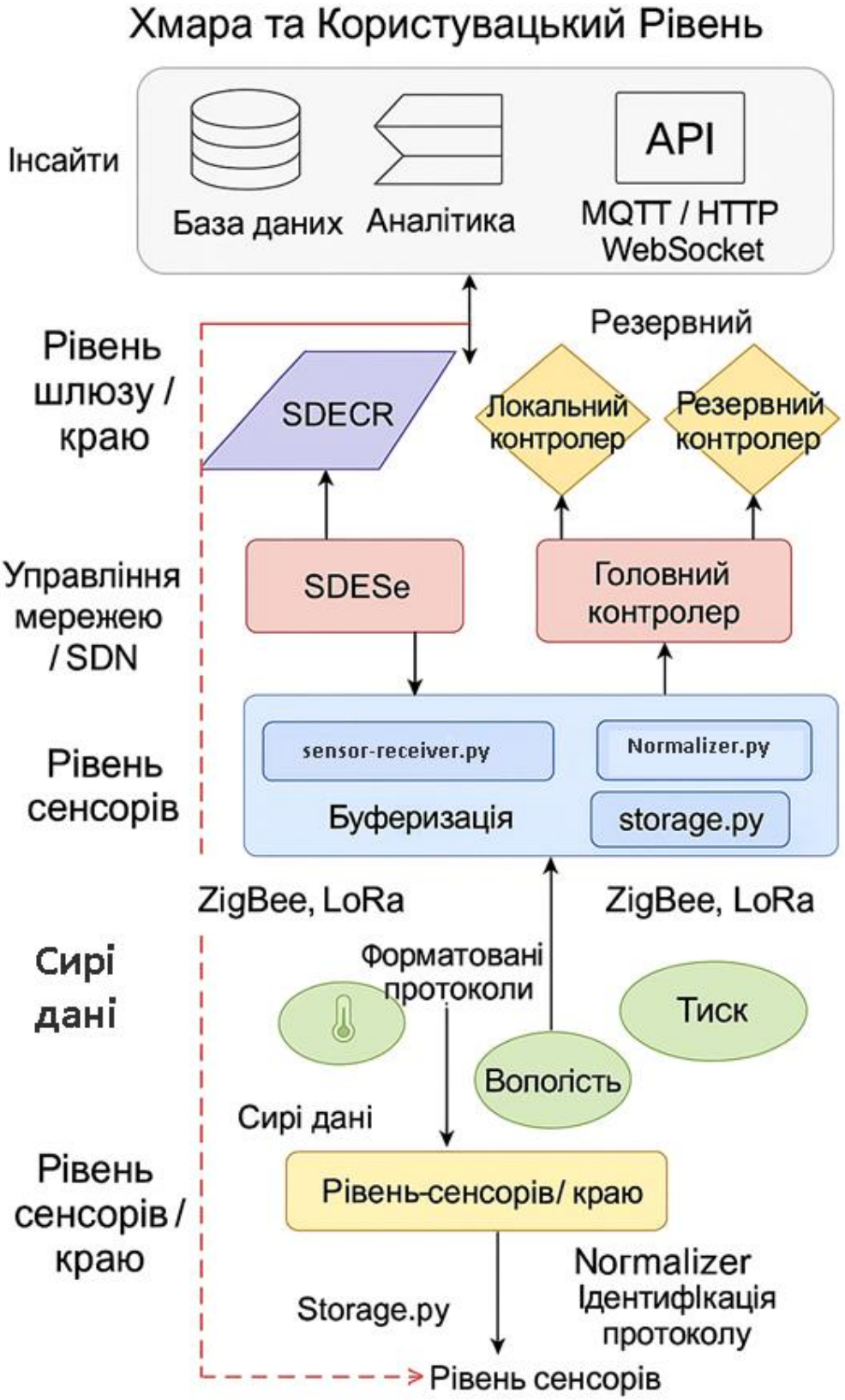
44. Трач М., Осолінський О. Концепція програмно керованої модульної архітектури IoT. Інтелектуальні інформаційні технології в прикладних дослідженнях : тези доп. студентської наук.-практ. конф. (м. Тернопіль, 27–29 травня 2025 р.). Тернопіль, 2025. С.319-321.

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

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

Додаток А

Концепція програмно керованої модульної архітектури IoT



Додаток Б

Код основних модулів

main.py

```
from normalizer import normalize_message
from storage import init_db, store_data
from protocol_detector import detect_protocol_by_port

incoming_port = 502  # Example: ModbusTCP
protocol = detect_protocol_by_port(incoming_port)

raw_data = {"temperature": 23.1, "humidity": 45}
device_id = "sensor_auto_001"

init_db()
json_msg = normalize_message(protocol, raw_data, device_id)
store_data(json_msg)

print(f"Protocol auto-detected: {protocol}")
print("Data processed and stored.")
```

normalizer.py

```
import json
from datetime import datetime

def normalize_message(protocol: str, raw_data: dict, device_id: str) -
> str:
    normalized = {
        "timestamp": datetime.now().isoformat(),
        "device_id": device_id,
        "protocol": protocol,
        "payload": raw_data
    }
    return json.dumps(normalized)
```

normalizer.py

```
import json
from datetime import datetime
```

```
    def normalize_message(protocol: str, raw_data: dict, device_id: str) -
> str:
        normalized = {
            "timestamp": datetime.now().isoformat(),
            "device_id": device_id,
            "protocol": protocol,
            "payload": raw_data
        }
        return json.dumps(normalized)
    root@debian:/home/Trach# cat protocol_detector.py
    PORT_PROTOCOL_MAP = {
        1883: "MQTT",
        5683: "CoAP",
        502: "ModbusTCP",
        80: "HTTP",
        443: "HTTPS",
        4840: "OPC UA",
        8883: "MQTT_TLS",
    }
    def detect_protocol_by_port(port: int) -> str:
        return PORT_PROTOCOL_MAP.get(port, "Unknown")
```

storage.py

```
    import sqlite3

    def init_db():
        conn = sqlite3.connect('gateway_data.db')
        c = conn.cursor()
        c.execute("""CREATE TABLE IF NOT EXISTS data
                     (timestamp TEXT, device_id TEXT, protocol TEXT,
payload TEXT)""")
        conn.commit()
        conn.close()

    def store_data(json_data: str):
        import json
        data = json.loads(json_data)
        conn = sqlite3.connect('gateway_data.db')
```

```
            c = conn.cursor()
            c.execute("INSERT INTO data VALUES (?, ?, ?, ?)",
                      (data['timestamp'], data['device_id'], data['protocol'],
json.dumps(data['payload'])))
            conn.commit()
            conn.close()
```

Додаток В
Апробація отриманих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(IITAR-2025)

27-29 травня 2025 року

Тернопіль

2025

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

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

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

Склад організаційного комітету конференції

Керівництво оргкомітету

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

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

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

Члени програмного комітету

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

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

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

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

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

Члени оргкомітету

Андрій Івасечко, викладач кафедри інформаційно-обчислювальних систем і управління

Дмитро Дюг, викладач кафедри інформаційно-обчислювальних систем і управління

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

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

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

Трач Мар'ян
студент групи КН-43
maryantrach@gmail.com
Осолінський Олександр
к.т.н., доцент
oso@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

КОНЦЕПЦІЯ ПРОГРАМНО КЕРОВАНОЇ МОДУЛЬНОЇ АРХІТЕКТУРИ IOT

Сучасні тенденції розвитку Інтернету речей (IoT) орієнтовані на побудову гнучких, масштабованих, адаптивних систем, що здатні інтегрувати гетерогенні пристрої, сенсори та протоколи в єдину функціональну інфраструктуру. У такому контексті особливого значення набувають програмно керовані архітектури, що дозволяють централізовано або розподілено контролювати та конфігурувати компоненти IoT-середовища.

Розвиток IoT привів до збільшення кількості взаємопов'язаних пристроїв, які збирають та обмінюються величезними обсягами даних. Ці пристрої потребують ефективного управління, обробки даних у реальному часі та адаптації до зміни умов середовища. У зв'язку з цим виникла концепція програмно визначених архітектур, зокрема програмно визначених мереж (Software-Defined Networking, SDN), яка дозволяє динамічно керувати ресурсами та функціональністю IoT-систем, забезпечуючи при цьому гнучкість, масштабованість і безпеку [1].

Загалом, предметна область програмно керованої модульної IoT-архітектури охоплює ряд важливих аспектів, зокрема:

1. Використання абстракції апаратного рівня через API, що дозволяє приховати складність взаємодії з різнорідними пристроями та забезпечити стандартизований підхід до розробки додатків;
2. Розділення функціональних рівнів (апаратний, базове ПЗ, прикладний рівень) задля досягнення гнучкості та повторного використання компонентів;
3. Підтримку мультипротокольної взаємодії (OPC UA, MQTT, CoAP, CORAL-SDN, RPL) для забезпечення сумісності з різними IoT-платформами та пристроями;
4. Реалізацію інструментів керування та візуалізації (GUI, XML-конфігурація), що дозволяє спростити налаштування, моніторинг та усунення несправностей;
5. Можливість централізованого або розподіленого управління компонентами системи, залежно від потреб конкретного домену застосування;

6. Підтримку гнучкого розгортання та динамічного масштабування, що є критично важливим для сценаріїв з великою кількістю пристроїв та динамічними навантаженнями.

Перевагою таких архітектур є їхня здатність до адаптації у складних індустріальних середовищах, забезпечення QoS, безпеки та інтероперабельності. Програмно визначені шлюзи та IoT-одиниці можуть забезпечити швидке оновлення функціональності, підтримку нових протоколів без зміни апаратного забезпечення [2], [3].

Окрім цього, гнучкість модульної архітектури дозволяє застосовувати підхід орієнтований на мікросервіси, де кожна функціональна одиниця представлена окремим модулем з чітко визначеним інтерфейсом [4].

Таким чином, формування програмно керованих IoT-систем є ключовим напрямом у розробці інфраструктур розумного виробництва, розумних міст, адаптивної охорони здоров'я, енергетики, логістики, агропромисловості та інших доменів цифрової трансформації.

Концепція програмно керованої модульної архітектури IoT

Концепція базується на сучасних підходах до побудови динамічних, масштабованих та адаптивних систем, які можуть функціонувати в умовах гетерогенних технологій, змінного середовища та високих вимог до безпеки, швидкодії й доступності. Суть цієї концепції полягає в розділенні фізичних пристроїв, обчислювальних ресурсів, логіки управління та сервісів в єдину багаторівневу архітектуру, кожен рівень якої виконує окремі функції та має свої власні алгоритми взаємодії, управління та обробки даних.

Загальна ідея програмно керованої модульної архітектури (рисунок 1) полягає у тому, що більшість функціональних можливостей, які раніше були закладені у апаратні компоненти, тепер реалізуються через програмні засоби.

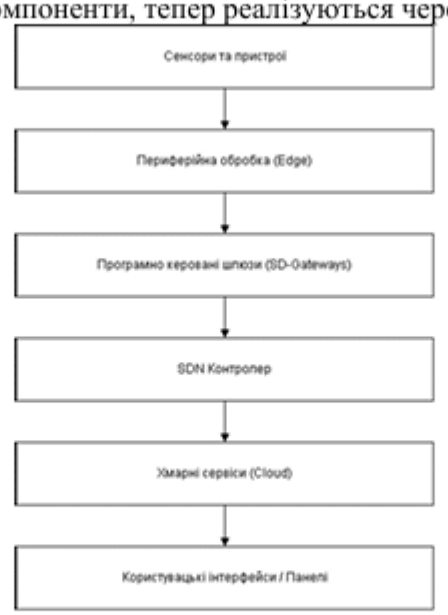


Рисунок 1 - Концепція програмно керованої модульної архітектури

Це дає змогу централізовано або розподілено управляти як самими пристроями, так і потоками даних, типами протоколів, сервісами аналітики та іншими компонентами IoT-середовища. Рішення, які приймаються в рамках цієї архітектури, дозволяють досягти більш високої продуктивності, адаптивності до змін у мережі, гнучкого розгортання нових пристроїв або сервісів без значних зусиль з боку адміністратора або розробника.

Архітектура складається з кількох рівнів, починаючи від фізичних пристроїв, що збирають дані з навколишнього середовища, до рівня хмарних сервісів, які здійснюють глибоку аналітику, візуалізацію та керування IoT-інфраструктурою.

Висновки

В результаті проведеного дослідження було сформовано концепцію програмно керованої модульної архітектури для IoT-систем, що враховує сучасні вимоги гнучкості, масштабованості та ефективності взаємодії між пристроями та сервісами.

Запропонована архітектура дозволяє чітко розмежовувати функціональні рівні: сенсорний рівень збору даних, рівень обробки на краю з протокольною нормалізацією, рівень мережевого управління на базі SDN-контролерів, рівень обробки подій, а також рівень візуалізації та аналітики.

Отримані результати можуть бути використані як основа для практичної реалізації гнучких IoT-рішень у сферах розумного сільського господарства, логістики, енергетики, міської інфраструктури та інших напрямків, де потрібна модульність, масштабованість та швидка реакція на події у фізичному середовищі.

Список використаних джерел

1. Jiang Z., Chang Y., Liu X. Design of software-defined gateway for industrial interconnection // *Journal of Industrial Information Integration*. 2020. T. 18. PP. 100130.
2. Kumar, S. S., & Agarwal, S. (2024). Rule based complex event processing for IoT applications: Review, classification and challenges. *Expert Systems*, 41(9), e13597.
3. Yuan, S., Wang, H., & Xie, L. (2021). Survey on localization systems and algorithms for unmanned systems. *Unmanned Systems*, 9(02), 129-163.
4. Li J., Gu W., Yuan H. Research on IoT technology applied to intelligent agriculture // Proceedings of the 5th International Conference on Electrical Engineering and Automatic Control, July 2016. Vol. 367. IEEE, 2016. PP. 1217–1224.