НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

"___"_____2016 р.

Дипломна робота

першого (бакалаврського) **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Аналіз засобів хмарного керування розумним будинком

Виконав: студент _4__ курсу, групи __ДА-22__
(шифр групи)

Жмудь Артем Володимирович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., Кірюша Б.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний професор, к.е.н. Семенченко Н.В. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____ст. викладач Британов О.А._____ _____
(науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський))

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування 7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2016 р.

ЗАВДАННЯ

на дипломний проект (роботу) студенту

Жмудю Артему Володимировичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)) Аналіз засобів хмарного керування розумним будинком

керівник проекту (роботи)) Кірюша Богдан Анатолійович, к.т.н., доцент

,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст

2. Строк подання студентом проекту (роботи) 08.06.2016

3. Вихідні дані до проекту (роботи) _____

Мікроконтролер NodeMCU на базі ESP8266

Датчик вологості

Датчик шуму

Сенсорна панель

Бюджет для аренди хмарної платформи – 0 грн

Доступ до мережі Інтернет через Wi-Fi

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Розглянути концепцію розумного будинку, інтернету речей та хмарних обчислень.

3. Розглянути існуючі API хмарного керування розумним будинком

5. Розробити робочу модель хмарного керування розумним будинком

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників, плакатів тощо)

1. Схема розумного будинку з контролером всередині – плакат.

2. Схема зв'язку робочої моделі з хмарним керуванням – плакат.

3. Схема підключення робочої моделі – плакат.

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Семенченко Н.В., професор, д.е.н.		

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Вивчення варіантів реалізації та вибір варіанту для розробки	28.02.2016	
4	Вибір способу реалізації	10.03.2016	
5	Розробка плану тестування	15.03.2016	
6	Розробка програмної моделі	25.03.2016	
7	Розробка опису системи	25.04.2016	
8	Тестування моделі	30.04.2016	
9	Оформлення дипломної роботи	31.05.2016	
10	Отримання допуску до захисту та подача роботи в ДЕК	08.06.2016	

Студент _____ А.В. Жмудь
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ Б.А. Кірюша
(підпис) (ініціали, прізвище)

_____

АНОТАЦІЯ

до бакалаврської дипломної роботи Жмудя Артема Володимировича

на тему: «Аналіз засобів хмарного керування розумним будинком»

Дипломна робота присвячена аналізу існуючих хмарних сервісів керування розумним будинком, а саме: аналіз існуючих найбільш популярних сервісів, які забезпечують безпечне та зручне управління всіма датчиками і пристроями розумного будинку.

Ціллю дипломної роботи є виявлення недоліків та переваг у роботі з розумним будинком окремих хмарних сервісів та реалізація тестової моделі на обраній платформі.

В роботі проведено аналіз хмарних сервісів. Були виявлені відмінності між розглянутими платформами та виділені сильні та слабкі сторони кожної з них.

В ході виконання дипломної роботи була побудована робоча модель з датчиками вологості, температури та шуму та зв’язком з хмарною платформою. Також був розроблений додаток для візуалізації даних з датчиків.

В результаті роботи було побудовано робочу модель розумного дому з використанням хмарних технологій для керування.

Загальний обсяг роботи – 70 сторінок, 18 рисунків, 9 таблиць, 17 посилань.

Ключові слова: Arduino, хмарна платформа, розумний будинок, MQTT, датчики.

АНОТАЦИЯ

К бакалаврской дипломной работе Жмудя Артёма Владимировича

На тему: «Анализ средств облачного управления умным домом»

Дипломная работа посвящена анализу существующих облачных сервисов управления умным домом, а именно: анализ существующих наиболее популярных сервисов, которые обеспечивают безопасное и удобное управление всеми датчиками и устройствами умного дома.

Целью дипломной работы является выявление недостатков и преимуществ в работе с умным домом отдельных облачных сервисов и реализация тестовой модели на выбранной платформе.

В работе проведен анализ облачных сервисов. Были выявлены различия между рассматриваемыми платформами и выделены сильные и слабые стороны каждой из них.

В ходе выполнения дипломной работы была построена рабочая модель с датчиками влажности, температуры и шума и связи с облачной платформой. Также было разработано приложение для визуализации данных с датчиков.

В результате работы была построена рабочая модель умного дома с использованием облачных технологий для управления.

Общий объем работы - 70 страниц, 19 рисунков, 9 таблиц, 17 ссылок.

Ключевые слова: Arduino, облачная платформа, умный дом, MQTT, датчики.

AN ABSTRACT OF THE THESIS OF

Bohdan Yevtukh for the degree bachelor of the computer science in NTUU “KPI”

Title: “Smart home control system. Security subsystem.”

Thesis is devoted to analysis of existing cloud-based smart home management services, namely, analysis of the existing most popular services that provide a safe and convenient control of all sensors and smart home devices.

The aim of the thesis is to identify strengths and weaknesses in dealing with smart home individual cloud services and the implementation of a test model of the selected platform.

The analysis of cloud services. Differences between these platforms and highlighted the strengths and weaknesses of each of them were identified.

In the course of the thesis with the working model of humidity sensors, temperature and noise was built due to a cloud platform. The application also has been designed to visualize the data from the sensors.

As a result of working model of smart home has been built with the use of cloud technology to manage.

The total volume of work – 70 pages, 19 pictures, 9 tables, 17 links.

Tags: Arduino, cloud platform, smart home, MQTT, sensors.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

SOAP – Simple Object Access Protocol

Розумний дім – будинок з мікроконтроллерами, що керують всіма або більшою частиною пристроїв всередині.

API – application programming interface

XML – Extensible Markup Language

MQTT – MQ Telemetry Transport

Publish–subscribe – шаблон проектування

Pub/Sub – Publish–subscribe

SDK – (software Development Kit) – набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми.

IoT – Internet of Things

HTTP – HyperText Transfer Protocol — протокол передачі гіпертексту

SDK – Software Development Kit

IaaS – Інфраструктура як послуга

ОС – операційна система

ВСТУП

Термін «розумний будинок» або «інтелектуальний будинок» використовується для позначення сучасних будинків і будівель, в яких інженерні, інформаційні системи і системи безпеки об'єднані в єдину і організовану комплексну інтелектуальну систему. Дана інтелектуальна система покликана забезпечувати більшу безпеку, а також найкращий комфорт мешканцям будинку. Як правило, основна причина установки систем розумного будинку полягає в підвищенні домашнього комфорту шляхом автоматизації рутинних завдань, таких як керування освітленням, клімат-контролем, системами мультимедіа і т.д.

Сьогодні, розумний будинок – це маленька держава. Як і у держави, у розумного дому є адміністративна служба, які обслуговують цю систему автоматично. Також в ній існує велика кількість автономнихуправлінь, які відповідають кожен за свій аспект привнесення щлагоди до «держави», а саме: система опалення та клімату, осітлення, водопостачання, контроль входу та виходу, пожежна служба та інші. Дуже легко провести паралель, з реально існуючими службами. Дії цих установ часто є неефективними через людський фактор, адже часто при екстренних ситуаціях приймати рішення потрібно за лічені секунди. Автоматизація управління різними сферами життя людини з розвитком технологій розумного будинку все частіше показує себе ефективніше, ніж ціла команда операторів, які цілодобово слудкують за показниками кожного окремого сектору або навіть кожного окремого будинку. Не можна не згадати вартість навчання персоналу і його заробітну плату. Набагато дешевше платити рахунки за електрику, яку споживають датчики, ніж цілому персоналу робітників.

Все гостріше постає питання доступності і легкості моніторингу на управління розумним будинком. Власник, або користувач, розумним домом повинен мати доступ до всіх систем, мати змогу керувати ними на відстані та

отримувати інформацію про стан кожної з них цілодобово. Він повинен бути впевненим, що ніхто без його відома не може отримати інформацію про його систему розумного дому, тим паче, мати змогу якось на неї вплинути. Великі компанії та сервіси, що надають послуги хмарних обчислень все частіше починають дивитися в сторону Интернету речей та систем розумного дому. Ставлячи на кін свою репутацію та йдучи на повідку о конкуренції, вони всіми силами намагаються зробити найзручніший, найшвидший та найбезпечніший сервіс управління розумним домом.

Як і у держави, у розумного дому є адміністративна служба, які обслуговують цю систему автоматично. Також в ній існує велика кількість автономнихуправлінь, які відповідають кожен за свій аспект привнесення щлагоди до «держави», а саме: система опалення та клімату, осітлення, водопостачання, контроль входу та виходу, пожежна служба та інші. Дуже легко провести паралель, з реально існуючими службами. Дії цих установ часто є неефективними через людський фактор, адже часто при екстренних ситуаціях приймати рішення потрібно за лічені секунди. Автоматизація управління різними сферами життя людини з розвитком технологій розумного будинку все частіше показує себе ефективніше, ніж ціла команда операторів, які цілодобово слудкують за показниками кожного окремого сектору або навіть кожного окремого будинку. Не можна не згадати вартість навчання персоналу і його заробітну плату. Набагато дешевше платити рахунки за електрику, яку споживають датчики, ніж цілому персоналу робітників.

Дана інтелектуальна система покликана забезпечувати більшу безпеку, а також найкращий комфорт мешканцям будинку. Як правило, основна причина установки систем розумного будинку полягає в підвищенні домашнього комфорту шляхом автоматизації рутинних завдань, таких як керування освітленням, клімат-контролем, системами мультимедіа і т.д.

1. КОНЦЕПЦІЯ РОЗУМНОГО БУДИНКУ. ІНТЕРНЕТ РЕЧЕЙ ТА ХМАРНІ ОБЧИСЛЕННЯ

1.1 Історія розвитку

Розумні будинки, як і більшість досягнень сучасної техніки, початково з’явилися на сторінках фантастичних оповідань. Але матеріалізовуватись ідея почала лише у XX-му сторіччі після широкого введення електрики у будівлях і розвитку інформаційних технологій. Перше повідомлення про віддалені прилади контролю можна віднести до розробки Ніколою Тесла дистанційного керування судами та транспортними засобами у 1898 році[3].

Електричні побутові прилади почали з’являтись між 1915 та 1920 рр. І одразу продемонстрували готовність суспільства замінити роботу домашнього персоналу дешевими механічними пристроями. Правда на той час, проблема енергозбереження при використанні нових технологій ще вирішена не була. Тому, певний час, новітні технологій були доступні лише дуже заможним людям.

Ідеї більш розвинені до понять сучасних систем автоматизації будинку були продемонстровані на ярмарках у Чикаго (1934) та Нью-Йорку. У “великому яблуці” трохи пізніше (1964-65), представили плани електрофікованих та автоматизованих приміщень . У решті-решт перший серйозний аналог розумного дому з’явився у 1966 році. Це була експериментальна система домашньої автоматизації – “домашній комп’ютер Эхо IV”. Його винахідник - Джим Сазерленд, інженер компанії en: Westinghouse Electric. Його технологія була приватним, некомерційним проектом. [4] Перші "дротові будинки" були зведені американськими винахідниками-любителями у 1960-х, але вони були суттєво обмежені можливостями тогочасних технологій.

Уперше термін "розумний будинок" був вигаданий Американською Асоціацією Housebuilders у 1984 році. Із винаходом мікроконтролерів, вартість на електроприлади швидко падала. Ця ж установа зазначила, що таке помешкання відмінне від звичайного своєю здатністю забезпечувати продуктивне та ефективне використання робочого та житлового середовища.

За цим, віддалені інтелектуальні технології керування були прийняті будівельною промисловістю, яка поступово почала вводити їх не лише у бізнес установах, але і у домашніх помешканнях. Під час активної домашньої автоматизації 90-х років інформатика та телевізійні системи були поєднані для підтримки інтелектуальних можливостей приміщень. У 1995 році винахідники технологій Java оголосили одним із основних призначень даної технології – “збільшення інтелекту побутових приладів”.

Сьогодні технології дозволяють збирати домашню автоматику покомпонентно: обирати лише ті функції розумного будинку, які дійсно потрібні користувачу. Тепер новітні технології керування приміщенням з’являються щодня. Навіть речі, котрі раніше розглядалися лише як красиві предмети інтер’єру тепер можуть виконувати ряд мультимедійних або побутових функцій.

1.2 Інтернет речей та хмарні обчислення

В даний час більшість систем розумного будинку не володіють функцією віддаленого управління через Інтернет. Тим часом мобільні пристрої з постійним доступом до мережі стали сьогодні буденним явищем, вони є практично у кожного.

У 1999 році засновник дослідницького центру Auto-ID Center в Массачусетському технологічному інституті Кевін Ештон запропонував термін Internet of Things (Інтернет речей). Його суть полягає в тому, що речі нового

покоління будуть не тільки «розумними», а й об'єднаними в мережу - Інтернет речей. [1] Концепція передбачає, що такі пристрої як смартфони, планшети, телевізори, різні датчики і керовані пристрої, що мають бездротові модулі Wi - Fi і Bluetooth, зможуть взаємодіяти між собою і користувачами за допомогою цих бездротових модулів. У зв'язку з масовим поширенням мобільних пристроїв, відповідних концепції Інтернет речей, стало можливим віддалене управлінням своїм розумним будинком. Очевидні переваги при наявності функції віддаленого управління системами розумного будинку: - Головна перевага - це звичайно ж безпека. При знаходженні мешканців за межами свого будинку або квартири можливо віддалене спостереження за допомогою камер за ситуацією або віддалений моніторинг в будинку шляхом відстеження стану різних датчиків, які використовуються в системах безпеки (пожежні датчики, датчики відкриття / закриття дверей і т.д.). Крім того, для тих, хто часто забуває вимкнути світло або якісь прилади, дана функція буде дуже корисною.

Основна ж перевага - це підвищення комфорту користувачів розумного будинку. Часто в керуючих системах розумного будинку використовують сценарії з управління світлом і теплом, коли вся робота здійснюється в автоматичному режимі. Найчастіше деякі користувачі вважають за краще обходитися без таких сценаріїв. І при наявності функції віддаленого управління користувач, наприклад, може сам при підході до свого будинку включити необхідні йому пристрої (увімкнути освітлення, побутові прилади, а також заздалегідь включити опалення або кондиціонер). Здійснення функції віддаленого доступу можливо за допомогою застосування хмарних обчислень, коли користувачі забезпечуються повсюдним доступом до мережевих обчислювальних ресурсів, сервісів і додатків. Існує кілька моделей хмарних обчислень.

Стосовно до розглянутого в роботі варіанту віддаленого управління системами розумного будинку більше підходить модель SaaS (програмне

забезпечення як послуга). Дана модель має на увазі надання клієнту доступу до програмного забезпечення через Інтернет. Основна перевага моделі SaaS для кінцевого користувача полягає у відсутності необхідності встановлення та оновлення програмного забезпечення, також йому не потрібно піклуватися про працездатність обладнання, на якому функціонує додаток.

При застосуванні хмарних обчислень в системах розумного будинку можливі два варіанти. У першому випадку контролер (сервер) для управління пристроями розумного будинку може бути розташований не в самому будинку (цю функцію візьме на себе хмара), завдяки чому управління системами розумного будинку може здійснюватися звідки завгодно при наявності доступу до Інтернету. При другому варіанті (рис. 1) контролер може розташовуватися в будинку, але при цьому через хмару буде забезпечуватися тільки віддалене управління - все програмне забезпечення буде встановлено на хмарному сервері.

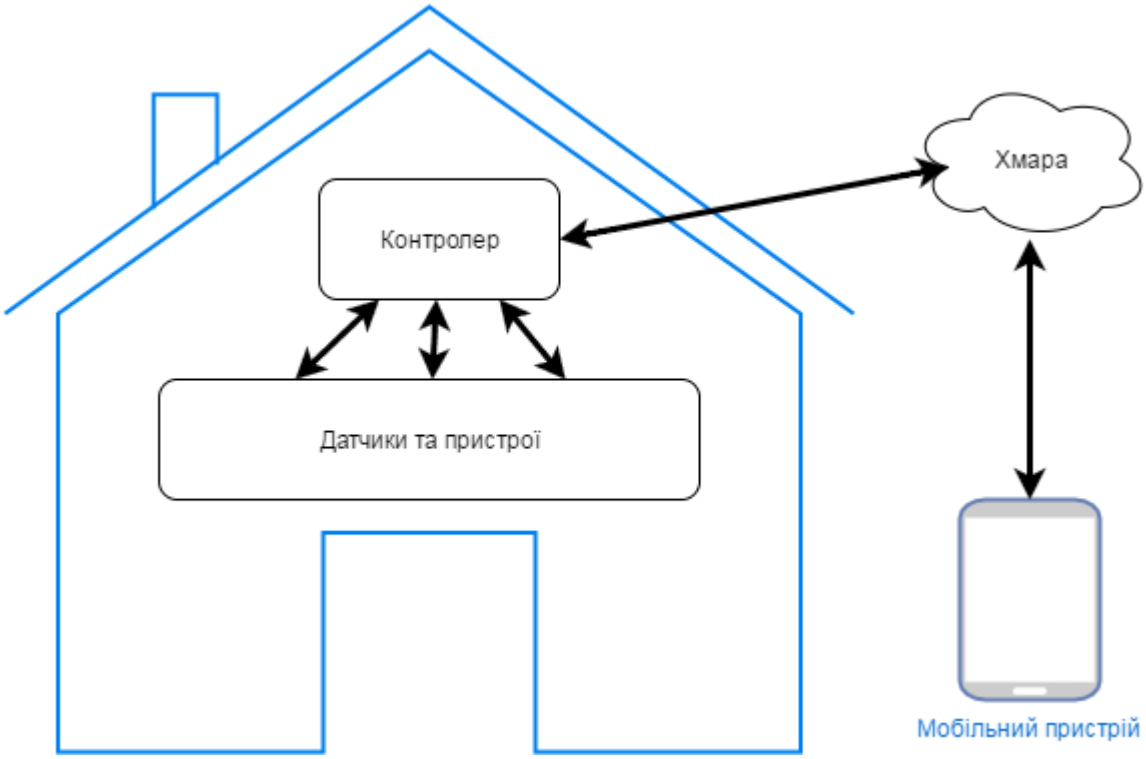

Рис. 1.1 - Контролер розташовуваний в будинку

Крім того, у другому випадку від домашнього контролера буде вимагатися тільки функція для забезпечення модулів розширення доступу до Інтернету, що в свою чергу зменшує вимоги до технічних характеристик контролера. Також в разі впровадження віддаленого управління вже в існуючу систему розумного будинку не буде потрібно заміни ніякого обладнання, досить буде лише забезпечити доступ контролера до хмарного серверу.

Безпосередньо віддалене управління системами розумного будинку можливо здійснювати або через веб-браузер, або через спеціальне мобільний додаток.

Варто відзначити ще одну важливу деталь. Багато сучасні пристрої, що використовуються в розумному будинку, як кінцеве обладнання, так і керуючі пристрої, працюють по своїх власних протоколах передачі даних і, крім того, можуть взаємодіяти з Інтернет-сервісами тільки через свої API. Тому часто немає можливості розширити систему розумного будинку, наприклад, яким-небудь розумним холодильником, або ж додати до неї пристрої, що працюють за іншими протоколами передачі даних. Однак за допомогою хмарного сервісу, який буде надавати загальний інтерфейс управління всіма системами, а різні пристрої будуть взаємодіяти між собою через хмару, з'являється можливість використання пристроїв від різних виробників з різними протоколами передачі даних. В результаті застосування хмарних технологій в системах розумного будинку дозволить зробити їх набагато більш гнучкими, а також дозволить скоротити витрати на обслуговування і розширення системи.

1.3 Протокол взаємодії хмарного сервера з пристроями розумного будинку

1.3.1 XML/Soap

Для успішної взаємодії хмарного сервера з пристроями розумного будинку ці обидва компонента повинні «розмовляти» один з одним на одній мові.

Одним із рішень в даному випадку є обмін даними через XML-повідомлення. Одним з протоколів, що використовують XML для обміну даними, є SOAP (від англ. Simple Object Access Protocol - простий протокол доступу до об'єктів). Основною перевагою використання SOAP є те, що він здатний забезпечувати безперервну взаємодію веб-сервісу з пристроями, що працюють по різних протоколах передачі даних.

Решта переваг застосування формату SOAP перед іншими форматами для передачі даних:

- кодувати в XML структури даних з використанням SOAP так само легко, як і дані простих скалярних типів;
- При використанні SOAP-повідомлень надаються додаткові інструменти, що дозволяють легко додавати, наприклад, функції забезпечення безпеки або трасування;
- Є набори інструментів SOAP для різних мов програмування. [2]

1.3.2 MQTT

MQTT (Message Queue Telemetry Transport) - спрощений мережевий протокол, що працює поверх TCP / IP. Використовується для обміну інформацією між пристроями за принципом publish–subscribe.

Перша версія протоколу була розроблена доктором Енді Станфорд-Кларком (IBM) і Арлен Ніппер (Arcom) в 1999 році і опублікована під роялті-фрі ліцензією. Специфікація MQTT 3.1.1 була стандартизована консорціумом OASIS в 2014 році.

- Шаблон проектування publish–subscribe зручний для більшості рішень з датчиками. Дає можливість пристроям виходити на зв'язок і публікувати
- Простий у використанні. Це програмний блок без зайвої функціональності, який може бути легко вбудований в будь-яку складну систему;
- повідомлення, які не були заздалегідь відомі або визначені;
- Легкий в адмініструванні;
- Знижено навантаження на канал зв'язку;
- Робота в умовах постійної втрати зв'язку або інших проблем на лінії;
- Немає обмежень на формат переданого контенту.

MQTT вимагає брокера повідомлень.(Рис.2) Брокер несе відповідальність за поширення повідомлень зацікавленим клієнтам на основі теми повідомлення.

MQTT визначає методи (іноді звані як дієслова), щоб вказати бажану дію, яка виконуватиметься на ідентифікованому ресурсі. Цей ресурс видає, колишні дані або дані, які генеруються динамічно. Це залежить від реалізації сервера. Часто, ресурс відповідає назві файла або виходу виконуваного файлу, розміщеного на сервері. Це значно знижує складність алгоритму при обробці даних, адже можна не переписувати дані двічі, а перевикористовувати строки. Це визначається прапором Retain, який передається разом з повідомлення. Це корисна практика для того, щоб клієнт не втратив дані, якщо раптово не встих їх отримати через нестабільну мережу.

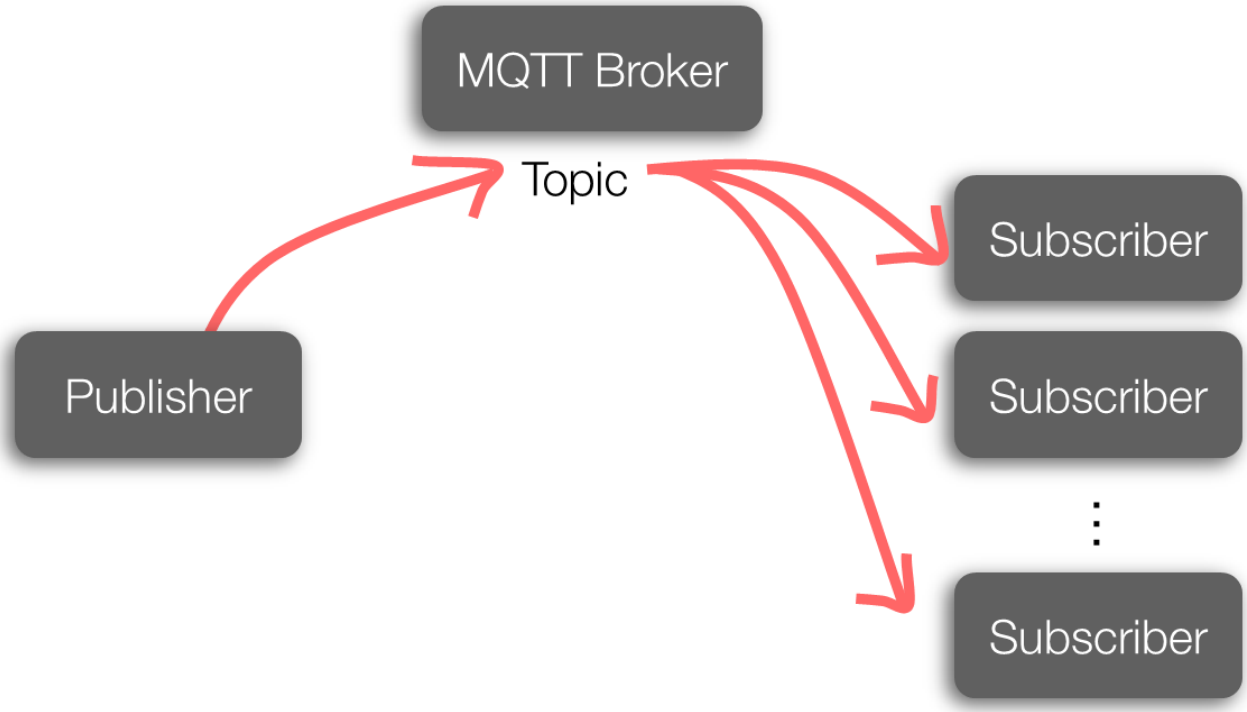

Рис. 1.2 – Схема MQTT брокера[9]

Є кілька відкритих брокерів MQTT:Emqttd, ActiveMQ, Apollo, Mosquitto, RabbitMQ та інші. Вони розрізняються за своїм набором функцій, і деякі з них реалізувати додаткові можливості поверх стандартного MQTT.

1.4 Висновки до розділу 1

За результатами аналізу історії розвитку та кінцевої стадії концепції розумного дома та методів передачі даних можно зробити висновок, що історія розвитку пройшла довгий та повний оновлень шлях, знайшовши найоптімальнішу концепцію роботи. Але і тут є куда розвиватись. Найбільші сервіси надання хмарних платформ все більше цікавляться інтернетом речей, про їх нововведення та вектори розвитку піде мова у наступному розділі.

2. АНАЛІЗ ІСНУЮЧИХ API ХМАРНОГО КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ

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.

2.1 AWS IoT (Amazon)

Платформа AWS IoT забезпечує підключення пристроїв до сервісів AWS і інших пристроїв, захист даних і безпеку взаємодії, обробку даних пристроїв і дії з ними, а також взаємодію додатків з пристроями навіть за відсутності підключення до Інтернету.[5]

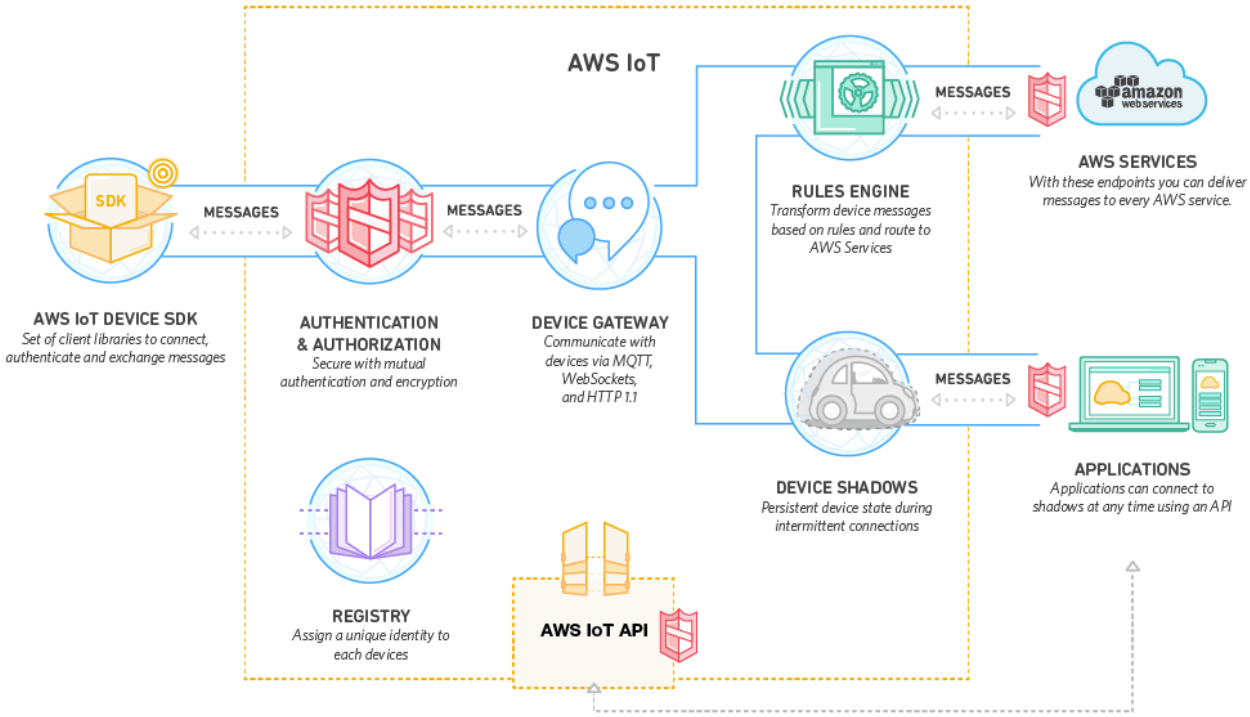

Рис 2.1 – Схема роботи AWS IoT [8]

2.1.1 AWS IoT SDK для пристроїв

Платформа AWS IoT надає SDK для простого і швидкого підключення апаратних пристроїв і мобільних додатків. AWS IoT SDK для пристроїв забезпечує підключення і аутентифікацію ваших пристроїв, а також обмін повідомленнями з платформою AWS IoT по протоколам MQTT, HTTP або WebSockets. Пакет SDK для пристроїв підтримує мови C, JavaScript і Arduino і містить клієнтські бібліотеки, керівництво для розробників і керівництво по перенесенню для виробників. Можна скористатися альтернативним SDK з відкритим вихідним кодом або написати власний SDK.

2.1.2 Шлюз пристроїв

Шлюз пристроїв AWS IoT забезпечує безпечну і продуктивну взаємодію пристроїв з платформою AWS IoT. Шлюз пристроїв підтримує обмін повідомленнями за допомогою моделі Publish–subscribe, що забезпечує взаємодію за схемами «один-до-одного» і «один-до-багатьох». В рамках схеми взаємодії «один-до-багатьох» AWS IoT дозволяє підключеному пристрою транслювати дані багатьом субскрайберам даної теми. Шлюз пристроїв підтримує протоколи MQTT, WebSocket і HTTP 1.1, і можна легко реалізувати підтримку пропрієтарних або застарілих протоколів. Шлюз пристроїв автоматично масштабується і може підтримувати більше мільярда пристроїв без необхідності виділення додаткової інфраструктури.

2.1.3 Аутентифікація і авторизація

Платформа AWS IoT забезпечує взаємну аутентифікацію і шифрування між усіма точками підключення. Таким чином будь-який обмін даними між пристроями і AWS IoT відбувається тільки з перевіркою ідентифікації. AWS IoT підтримує метод аутентифікації AWS, званий SigV4, а також аутентифікацію на основі сертифікатів стандарту X.509. Підключення по протоколу HTTP можуть використовувати будь-який з цих методів.

Підключення по протоколу MQTT використовують аутентифікацію на основі сертифікатів. Підключення по протоколу WebSockets можуть використовувати метод SigV4. AWS IoT дозволяє використовувати сертифікати, сформовані самим сервісом AWS IoT, а також сертифікати, підписані обраним центром сертифікації. Можна прив'язати роль і / або політику до кожного сертифікату для надання авторизованого доступу пристроїв або додатків з можливістю відкликання дозволу доступу без необхідності працювати з пристроєм безпосередньо.

Можна створювати та використовувати сертифікати і політики пристроїв і управляти ними з Консолі або за допомогою API. AWS IoT також підтримує підключення з боку мобільних додатків користувачів за допомогою сервісу Amazon Cognito, який повністю забезпечує створення унікальних ідентифікаторів для користувачів ваших додатків і отримання тимчасових даних доступу AWS з обмеженими правами.

2.1.4 Реєстр

Реєстр встановлює ідентифікацію для пристроїв і дозволяє відстежувати метадані, такі як атрибути або можливості пристрою. Реєстр дозволяє унікальним чином ідентифікувати кожен пристрій відповідно до єдиного формату, не залежних від типу пристрою або його підключення. Він також підтримує використання метаданих, що описують можливості пристрою, наприклад реєстрацію датчиком температури за шкалою Фаренгейта або Цельсія.

Реєстр дозволяє зберігати метадані пристроїв без додаткової плати. Щоб уникнути видалення збережених в реєстрі метаданих, досить звертатися до відповідного запису реєстру або оновлювати її хоча б раз в 7 років.

2.1.5 Фреймворк правил

Движок правил дозволяє створювати додатки IoT для збору, обробки і аналізу даних, що генеруються підключеними пристроями, і виконання дій з ними в глобальних масштабах без необхідності керувати будь-якою інфраструктурою. Движок правил оцінює вхідні повідомлення, що публікуються в AWS IoT, перетворює і доставляє їх іншому пристрою або хмарному сервісу з урахуванням заданих бізнес-правил. Правило можна застосовувати до даних від одного або від багатьох пристроїв і виконувати на його основі одну дію або ж безліч паралельних дій.

Движок правил також може перенаправляти повідомлення в кінцеві точки AWS, в тому числі AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch і Amazon Elasticsearch Service з вбудованими можливостями інтеграції з платформою Kibana. Доступ до зовнішніх кінцевих точок можливий за допомогою сервісів AWS Lambda, Amazon Kinesis і Amazon Simple Notification Service (SNS).

Можна створювати правила за допомогою Консолі управління або писати їх, використовуючи SQL-подібний синтаксис. Правила можна розробити таким чином, щоб вони діяли по-різному в залежності від вмісту повідомлення. Наприклад, якщо показання температурного датчика перевищило деяке порогове значення, може запускатися правило для передачі даних в AWS Lambda. Правила можуть також передбачати облік інших даних в хмарі, наприклад даних, отриманих від інших пристроїв. Наприклад, можна задати виконання дії, якщо температура більше ніж на 15% перевищує середній показник п'яти інших пристроїв.

У движку правил доступні десятки функцій для перетворення даних, а за допомогою сервісу AWS Lambda можна розширювати їх число до нескінченності. Наприклад, при роботі з широким діапазоном значень можна

обчислювати середнє значення вхідних чисел. За допомогою правил можна також ініціювати виконання коду на Java, Node.js або Python в AWS Lambda, що відкриває надзвичайно гнучкі можливості потужної обробки даних від пристроїв.

2.2 Google Cloud Platform IOT Solutions

Хмарна платформа надає можливості, які можна використовувати в своїх інтересах для щоденних операцій:[6]

- Google Cloud Monitoring надає панелі моніторингу та оповіщення для хмарних додатків. На пристроях Linux можна встановити Cloud Monitoring agent, який є Stackdriver на основі системи агента. Крім того, можна використовувати API Monitoring Cloud з потрібники метриками.
- Google Cloud Logging збирає і зберігає журнали, таким чином, можна переглядати, шукати, фільтрувати і експортувати інформацію. Використання Cloud Logging може заощадити багато часу і зусиль у порівнянні з побудовою рішення на замовлення.
- Google Cloud audit logs охоплює адміністрацію та доступ до даних діяльності, пов'язаної з Cloud Platform, зберігаючи їх у незмінних журналах, які можуть бути використані для аудиту.

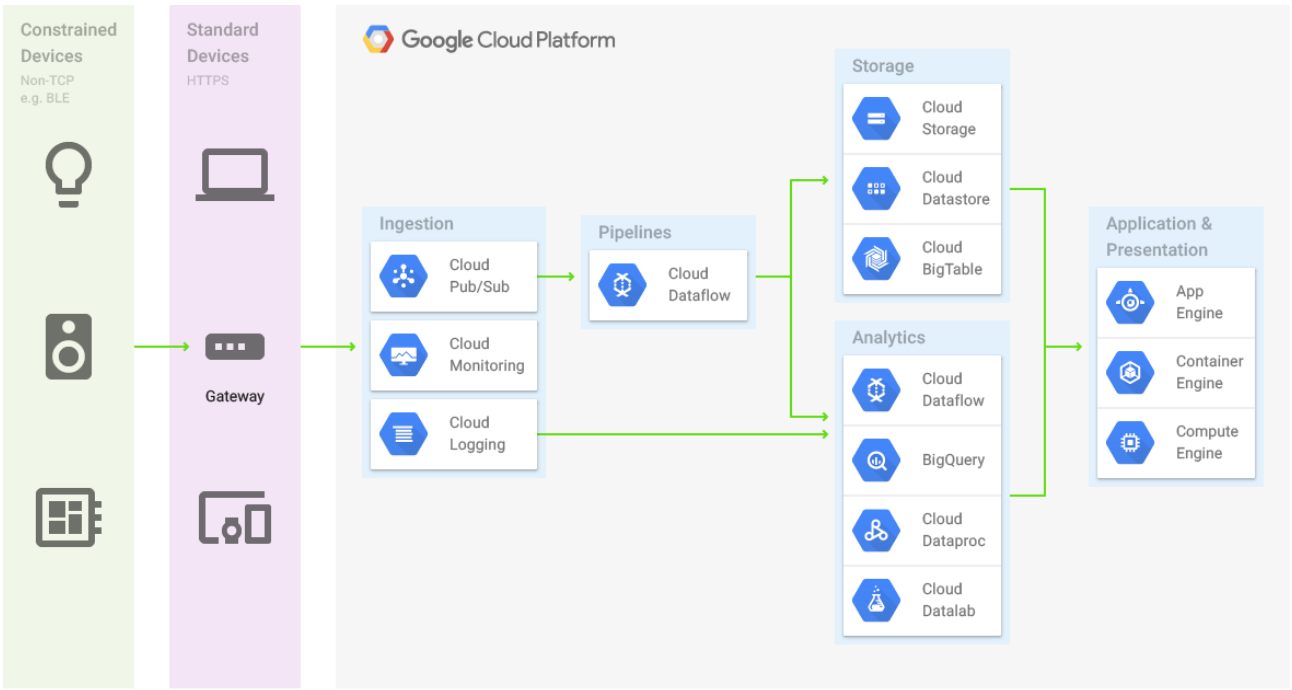

Рис. 2.2 - Різні етапи управління даними IoT в Google Cloud Platform[7]

2.2.1 Cloud Pub/Sub

В Google Cloud Pub / Sub створюючи теми для потоків або каналів, можна включити різні компоненти застосування, щоб підписатися на певні потоки даних без необхідності побудови індивідуальних абонентських каналів на кожному пристрої. Cloud Pub / Sub також спочатку підключається до інших сервісів Cloud Platform, допомагаючи підключити прийом даних, шлюзів і систем зберігання даних.

Cloud Pub / Sub може діяти як амортизатор і зрівнювач швидкості для обох вхідних потоків даних і зміни архітектури додатків. Багато пристроїв мають обмежені можливості для зберігання і надсилання даних телеметрії. Платформа може використовуватися для обробки даних шипи, які можуть

виникнути, коли багато пристроїв реагують на події в фізичному світі, і буферизувати ці шипи, щоб допомогти ізолювати їх від додатків моніторингу даних.

На додаток до стандартних API, HTTPS REST, Cloud Pub / Sub також підтримує gRPC, продуктивну, з відкритим вихідним кодом. Вона забезпечує більш високу пропускну здатність бінарних відформатований повідомлень. На діаграмі[Рис.2.3] показані результати тестування продуктивності з використанням Java-клієнта повідомлень публікація 50kb при максимальній пропускній здатності від одного компьютера, використовуючи 9 gRPC каналів.

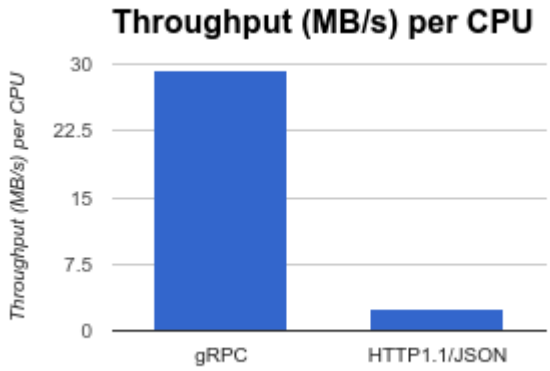

Рис.2.3 – Порівняльна характеристика gRPC та HTTP[7]

2.2.2 Зберігання даних

Дані з фізичного світу приходить в різних формах і розмірах. Хмарна платформа пропонує широкий вибір рішень для зберігання даних з неструктурованих згустків даних, таких як зображення або відео потоків, структурованого зберігання показників пристроїв або транзакцій.

Стан пристрою в загальному випадку може бути змодельований як набір пар ключ-значення. Деякі пристроїможуть бути підключені безпосередньо до

апаратних засобів. Інший стан пристрою може існувати на рівні додатків. Часто цінно, навіть необхідно, для іншого програмного забезпечення, такого як мобільний додаток або веб-сайт, прочитати або змінити останній стан цього пристрою. З огляду на те, що IOT пристрої можуть провести деякий час в малопотужний режим сну і може існувати на особливо ненадійних мережах, часто буває корисно відобразити деякі стани для пристрою з хмарою. Таким чином, дані стану можуть бути зроблені доступними навіть тоді, коли самі пристрої тимчасово відсутній.

Firebase добре підходить для підтримки локальної копії стану[Рис.2.4]. Вона дозволяє виставити уявлення поточного стану пристрою через Firebase ключі перегляду. Firebase клієнтські бібліотеки полегшують видавання інформації різним спостерігачам.

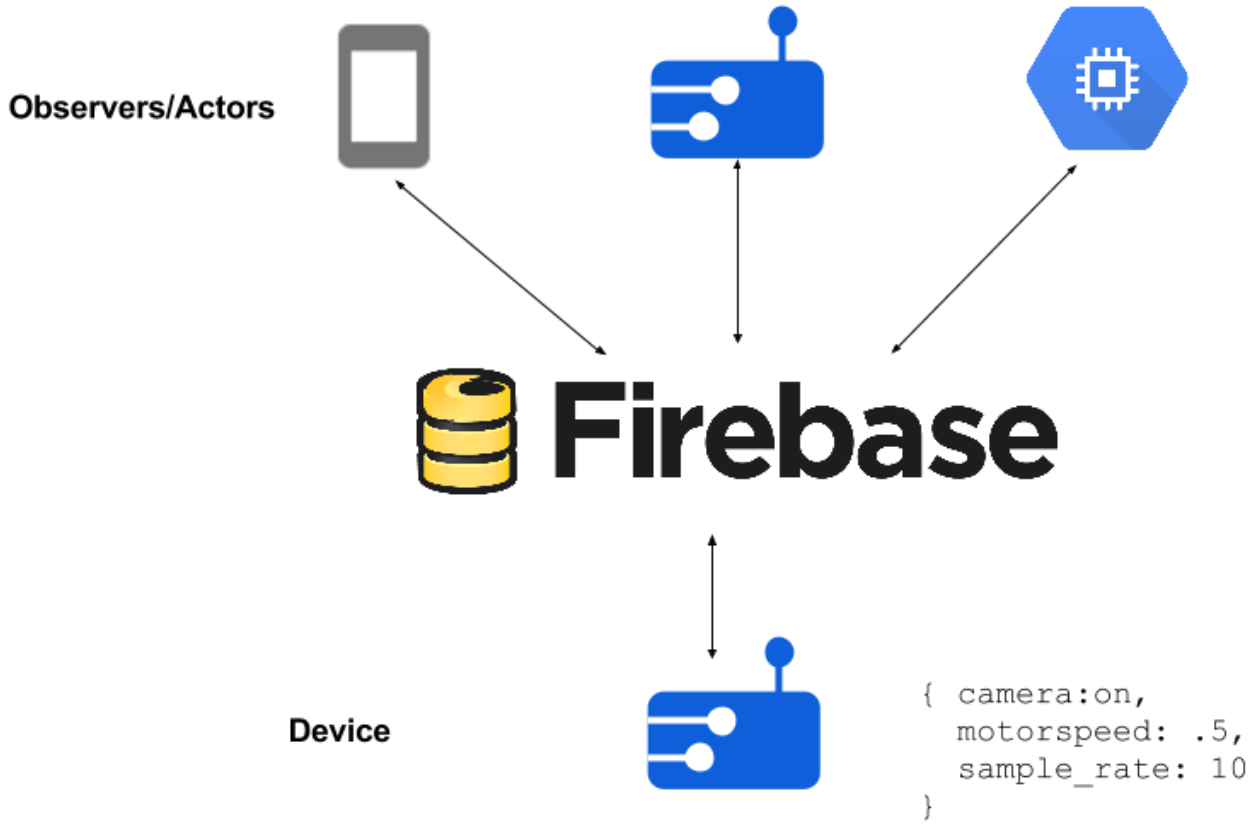

Рис 2.4 – Схема роботи Firebase[7]

2.3 Smart Home Cloud API (Samsung Smart Home)

Smart Home Cloud API надає методи для управління і моніторингу Samsung Smart Home пристроями. За допомогою Smart Home Service Control, додаток може з'єднатися з різними пристроями, а також надати розширені послуги для своїх клієнтів.[7]

Smart Home Service Control працює через інтеграцію хмара-хмараміж клієнською хмарою і Smart Home Cloud.

Samsung надає кілька REST API для партнерів, щоб вони могли інтегрувати свою систему до Samsung Smart Home Cloud. Тіло REST API буде використовувати стандартний формат JSON документ. Таким чином, розробники-партнери повинні розуміти документ в форматі JSON, названий Smart Home Data Model.[Рис.2.5]

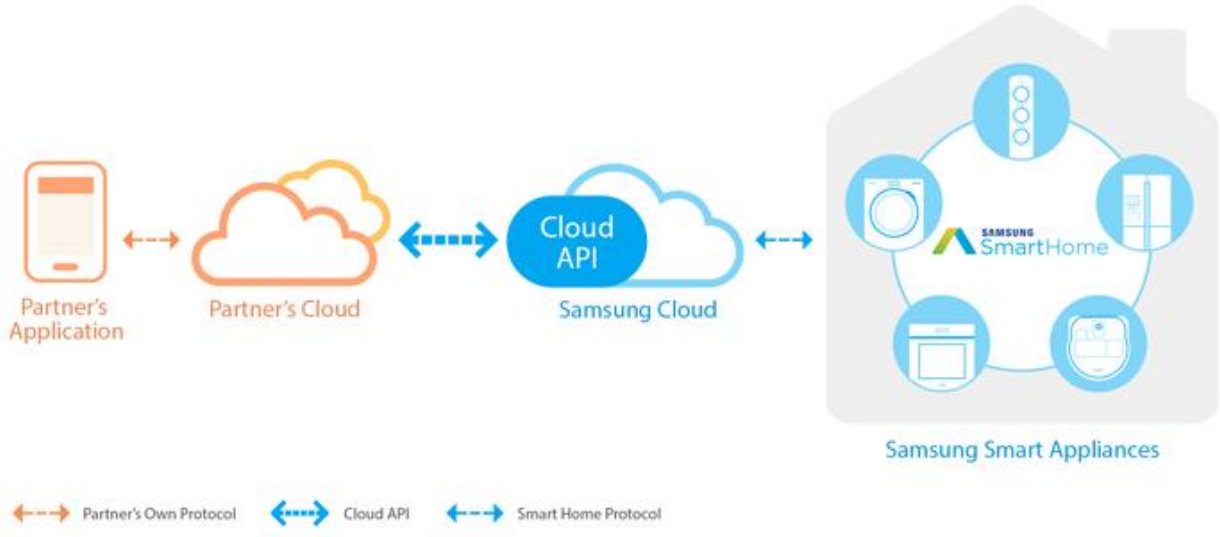

Рис.2.5 – Специфікації Smart Home Cloud API[6]

2.3.1 Модель інтеграції

Smart Home Data Model є структурою даних в форматі JSON для подання Samsung Smart Home пристроїв, як холодильники, пральні машини, кондиціонери, очищувачі повітря, робот пилососи, сушарки та печі.

Кроки по інтеграції користувачів (партнерів) Smart Home Cloud:

- Authentication. Для того, щоб з'єднатися з пристроями Samsung, ваша система повинна стежити за процесом авторизації аккаунта Samsung. Це означає, що ви можете спілкуватися з Samsung Home Cloud, якщо він має дійсний маркер, отриманий за допомогою процесу OAuth.
- Discovery. Партнер може отримати список всіх розумних домашніх пристроїв, зареєстрованих для конкретного користувача. У ньому міститься докладна інформація, така як тип, назва, модель, версію і ресурси.
- Sensing. Партнер може запросити інформацію про стан конкретного пристрою. Sensing API забезпечує найнижчий статус ресурсу конкретного пристрою.
- Subscription. Партнер може зареєструвати повідомлення про зміни в Smart Home пристрою, а потім отримувати дані в режимі реального часу.
- Notification. Якщо Samsung Home Cloud виявляє зміна в стані пристрою, вона відправляє дані повідомлення і статус партнерам.
- Control. Партнер може керувати певним пристроєм за допомогою API керування в Smart Home Cloud.
- Unsubscription. Партнер може відмовитися від отримання повідомлень про зміну для пристрою. Потім партнер не отримає будь-яких даних повідомлення від пристрою.

2.4 Не висвітлені хмарні платформи

Було проаналізовано декілька хмарних платформ для керування розумним будинком. Деякі представники хмарних платформ не були висвітлені в дипломній роботі через брак інформації на сайті, незрозумілий тип інтеграції, недостатню документацію або через чисто суб’єктивну негативну оцінку автора даної дипломної роботи [Таблиця 2.1].

Таблиця 2.1 – 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

2.5 Порівняльна характеристика

Порівняльна характеристика розглянутих платформ буде проводитися по наступним критеріям:

1. Безпека передачі даних
2. Можливості розширення
3. Різноманітність пристроїв, які можуть бути підключені
4. Моніторинг та аналіз даних

Критерії розташовані в довільному порядку. Важливість (вага) кожного з критеріїв буде встановлена далі.

Було розглянуто три хмарні платформи. Для подальшої зручності, вони будуть названі тільки шифром компанії, яка надає послуги, а саме: Amazon IoT, Google IoT, Samsung IoT.

2.5.1 Безпека передачі даних

Основні запропоновані методи забезпечення безпека інформації, що передається з датчиків та з клієнтських програм управління/адміністрування – це SSL з’єднання, Аутентифікація сессії з’єднання через логін та пароль, унікальний ідентифікатор кожного пристрою[Таблиця 2.2].

Таблиця 2.2 – Оцінка безпеки передачі даних

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

Google IoT не виділяє унікальний ідентифікатор на кжен датчик. Натомість, Google видяє ідентифікатор лише на групу пристроїв, які об’єднані одним маршрутизатором.

Samsung IoT має унікальну систему Samsung OAuth, яка є єдиним аккаунтом на всі Samsung пристрої. Детальну інформацію можна отримати лише в разі доступу до цього аккаунту, маючи пристрій Samsung, тому ніякої додаткової інформації добути не вдалось.

2.5.2 Можливості розширення

Amazon IoT

Стартовий (безкоштовний) пакет послуг включає в себе відразу всі складові: базу даних, розрахункові потужності та канал зв’язку. Ціни на Amazon IoT відкривають цілий ряд можливостей для зниження витрат. Цінова модель, що припускає оплату за фактом використання, дозволяє планувати розширення або підвищення вимог, пов'язане з сезонністю, дає можливість планувати бюджет відповідно до потреб бізнесу. Є можливість окремо розширити розрахункові потужності, фізичну пам’ять та ширину каналу передачі данних, відповідно до потреб користувача.[8]

Google IoT

Стартовий (безкоштовний) пакет послуг надає доступ до основних складових, а саме: базу даних, розрахункові потужності та канал зв’язку. Додатково можна підключити ряд API, які виконують різні функції. Наприклад:

- Робота з Big Data
- Логування
- Machine Learning
- Networking – DNS засоби, балансування загруженості мережі

Всі вони є достатньо дорогими, та потребують знання прикладних областей. Тобто звичайному користувачу все рівно потрібна допомога фахівця щоб правильно налаштувати ці додаткові системи, але з іншої сторони це може значно скоротити час розробки та налаштування системи. Додаткові послуги оплачуються до моменту старту користування.

Samsung IoT

Пакет послуг є платним від початку. Всі послуги надаються відразу та розширюються автоматично, залежно від виходу нових версій та оновлень платформи Samsung IoT. Безкоштовної безплатної версії не надається.

Таблиця 2.3 – Оцінка можливості розширення

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

2.5.3 Різноманітність пристроїв, які можуть бути підключені

Amazon IoT

Amazon IoT надає SDK для Embedded C, JavaScript та специфічно для Arduino Yún. Багато мікроконтролерів Arduino підтримують прошивку через Embedded C, тому це відкриває можливості інтеграцію на велику кількість

пристроїв, які можно власноруч прошити та налаштувати відповідно то потреб користувача.

Google IoT

Технологія Google Cloud Pub/Sub надає можливість підключити датчики та мікроконтролери через пртокол зв’язку gRPC, який можна встановити на велику кількість мікроконтролерів, в тому числі і на Arduino NodeMCU.

Samsung IoT

Платформа підтримує зв’язок лише з продуктами, випущеними компанією Samsung. Це різко знижує можливість бюджетної реалізації тестування платформи.

2.5.4 Моніторинг та аналіз даних

Amazon IoT

Платформа дозволяє створювати так звані «тіні» датчиків і пристро’в у хмарі. Це по суті віртуальна модель пристрою, в яку записуються його показники або віддаються команди. Також є можливість створення приватних та публічних ключів для передачі інформації по захищеному каналу.

Є можливість створення «ролей» - це так звані тригери, які можна запрограмувати на певні події, такі як критичні показники датчиків, сигнали тривоги та прості команди з клієнтської частини системи. Через вікрите API та SDK можна витягувати інформацію на клієнтську частину системи через технологію MQTT, що дає змогу зручно зчитувати дані з датчиків і відображати на пристроях керування, таких як телефон, сайт, десктоп та інших.

Google IoT

Платформа надає велику кількість засобів для аналізу даних. Виведення графіків, відстежування показників у часі та обробку Big Data. Також є інтегрована нейронна сітка, яку можна використовувати для автоматичного керування пристроями за допомогою засобів штучного інтелекту. Ця технологія, проте, знаходиться у бета режимі і дозволяє єфективно працювати лише при великому наборі навчальних даних.

Samsung IoT

Особливістю платформи є надання спеціалізованого програмного забезпечення на базі Android, які можна легко встановити на Samsung пристрої та отримувати інформацію з хмари без написання спеціалізованого програмного забезпечення. Є система нотифікацій, які можна налаштувати по пріоритетам, підписатися на- / відписатися від певних подій, відображати показники за великий проміжок часу. Нажаль інші засоби моніторингу даних дослідити не вдалося через брак інформації, яка надається неліцензійованим користувачам платформи.

2.6 Висновки до розділу 2

У ході дослідження біли проаналізовані платформи хмарного керування розумним будинком, такі як: AWS IoT (Amazon), Google Cloud Platform IOT Solutions, Smart Home Cloud API (Samsung Smart Home) за чотирма параметрами:

1. Безпека передачі даних
2. Можливості розширення

3. Різноманітність пристроїв, які можуть бути підключені
4. Моніторинг та аналіз даних

В аспекті безпеки передачі даних всі три платформи виявилися однаково надійними, реалізуючи передові технології в шифруванні данних та аутентифікації користувачів та пристроїв.

В аспекті можливостей розширення домінує платформа Google IoT через її великий набір додаткових засобів для аналізу та моніторингу даних. Враховуючи те, що всі технології цієї платформи є платними і те, що можна легко знайти безкоштовні аналоги та впровадити їх до платформи Amazon IoT, то в даному аспекті немає чіткого лідера і все залежить від навичок розробника, часу та бюджету, які є в наявності у проекта. Samsung IoT в данному випадку подається з уже готовим набором інструментів, який важно кастомізувати під власні потреби.

В аспекті різноманітності пристроїв, які можуть бути підлючені Samsung IoT значно відстає через те, що він дозволяє підлючення тільки пристроїв, які були виготовлені компанією Samsung. На відміну від Amazon Iot та Google IoT, які надають розробникам більшу свободу у виборі потрібних датчиків, пристроїв, протоколів та контролерів.

В аспекті моніторингу та аналізу даних всі платформи надають потрібний і достатній набір інструментів для адміністрування та налаштування приладів та датчиків. Google IoT має потужні інструменти візуалізації даних, які в свою чергу можна легко імплементувати до Amazon IoT. Samsung IoT лише дає опис своїх засобів без можливості протестувати їх на тестових даних, що є великим недоліком, коли мова йде про вибір платформи.

3. РОЗРОБКА РОБОЧОЇ МОДЕЛІ ХМАРНОГО КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ

Було проаналізовано найпопулярніші та, потенційно, найнадійніші хмарні платформи для керування розумним будинком. Всі мають свою сильні та слабкі сторони, тому було вирішено розробити свою власну систему керування розумним будинком з урахуванням найкращих практик, запозичених з результатів аналізу хмарних сервісів.

3.1 Схема підключення робочої моделі розумного будинку

Схема побудована на базі мікроконтролера NodeMCU v3 LoLin з вбудованим Wi-Fi модулем ESP8266[Рис. 3.1]. Даний контролер, попри свою низьку ціну, є дуже зручним засобом розробки цифрових схем для розумного будинку.

Основні переваги NodeMCU:[11]

- Простота програмування;
- На основі Lua 5.1.4 (без дебагу, лише ОС модулы);
- Асинхронна івентова модель програмування;
- 40+ вбудованих модулів;
- Прошивка доступна з або без підтримки плаваючої точки (цілочисленні використовують менше пам'яті)
- Проект постійно оновлюється, як і його документація[10]
- Підтримка програмування на декількох мовах, в тому числі C

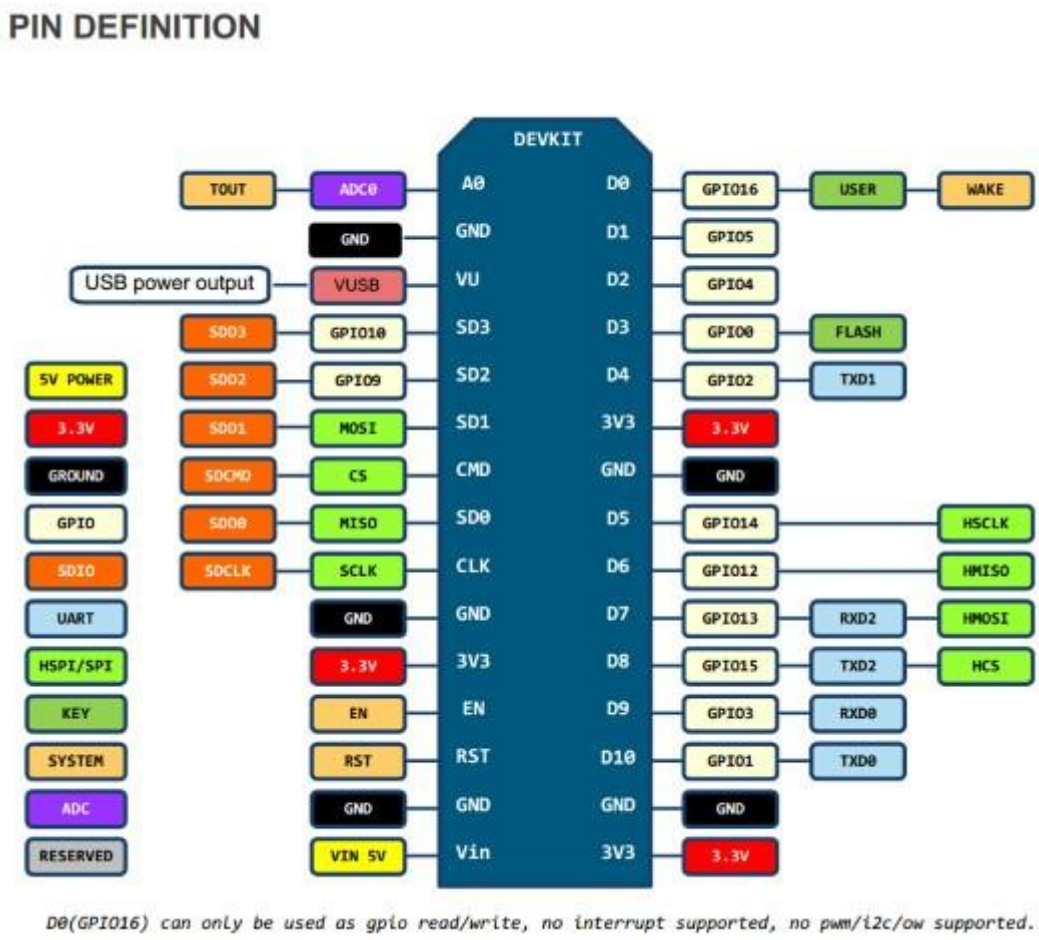

Рис. 3.1 – Схема мікроконтролера NodeMCU v3 LoLin[11]

Програмування виконується в середовищі розробки Arduino IDE на мові програмування C. Мікроконтролер підключається до комп'ютера через microUSB, що забезпечує його живленням. Також через microUSB виконується його програмування.

До тестової робочої схема були підключені наступні датчики:

- Датчик сили/моменту hex c227986
- Датчик температури та вологості повітря Aosong am2302
- Сенсорна панель з чотирма сенсорними кнопками
- Світлодіодна лампочка

Схема підключена показана на рисунку 3.2[Рис 3.2]

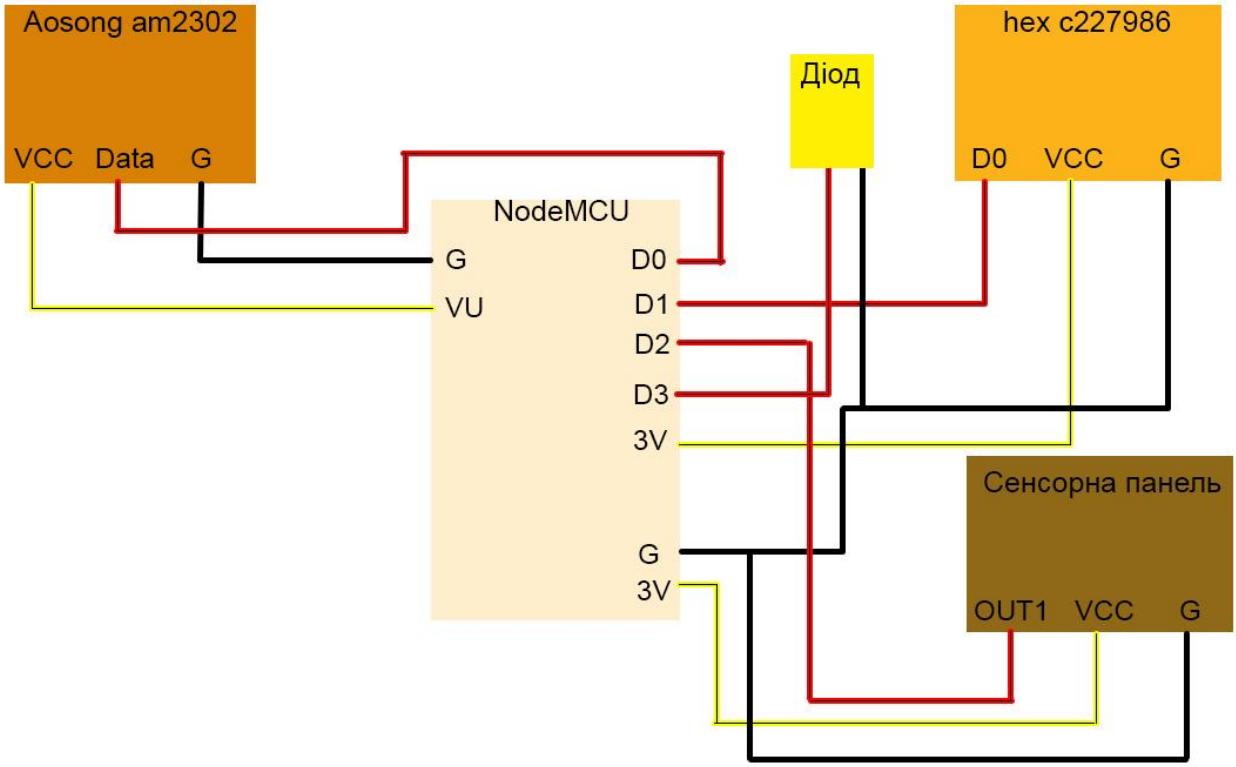

Рис. 3.2 – Схема підключення робочої моделі

3.2 Протокол передачі даних з датчиків

Протоколом передачі даних з датчиків було обрано протокол MQTT. Брокером для обміну інформації біло обрано хмарний сервіс CloudMqtt[12]. Пакет послуг в хмарному середовищі – Cute Cat[Рис. 3.3]. Сервіс виділяє 10 безкоштовних з'єднань на швидкості 10 Kbit/s, чого цілком вистачить для тестової моделі.

Передача даних с датчиків відбувається за наступною схемою: температура/вологість – 1 раз за 3 секунди, датчик шуму та сенсорна панель – 1 раз в 1 секунду (тільки при зміні значення).

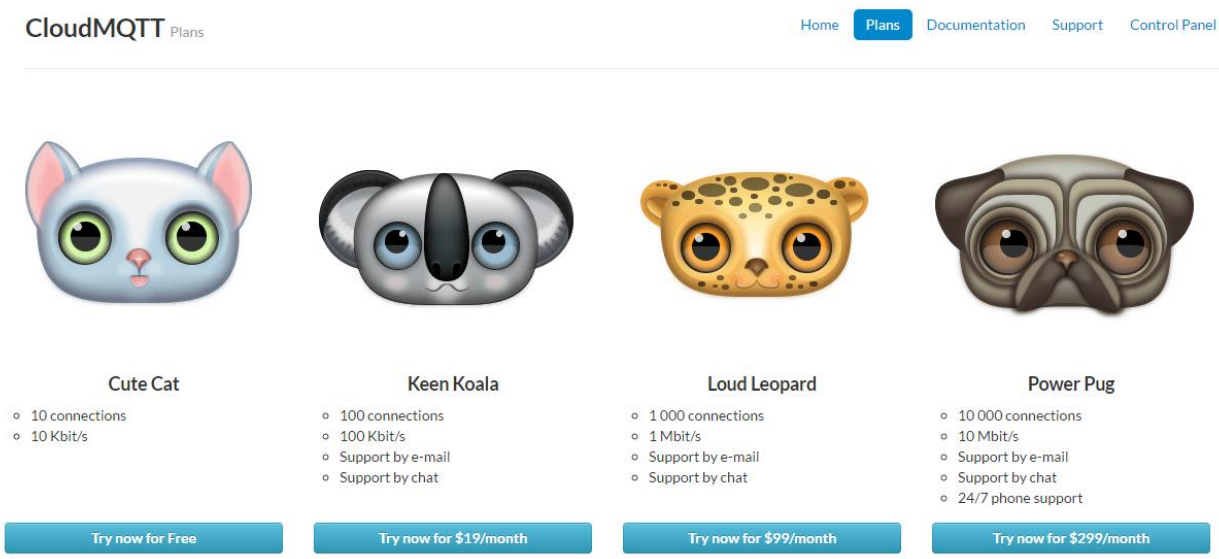

Рис. 3.3 – Бізнес-плани CloudMqtt [13]

3.3 Веб-застосунок для керування робочою моделью

Було вирішено об'єднати бізнес-логіку та сервер сайту на одному сервері через невелику загруженість робочої моделі. Був використаний сервер NodeJS[14] з підключеною бібліотекою Express[15]. Так як мова фронт-енд частини сайту була написана з використанням мови javascript, то вибір сервера вважається цілком обгрунтованим через швидкість написання та достатню швидкість для даного набору датчиків та можливого каналу зв’язку.

Через необхідність частого обміну інформацією між веб-сторінкою та сервером, був обраний протокол передачі даних socket.io[16]. Це забезпечує швидкий та зручний в обробленні даних зв’зок, який потребується для швидкого реагування на зміни показників датчиків.

У хмарному сервісі Amazon був виділений окремий віртуальний сервер EC2 на ОС Ubuntu, на який було встановлено веб-сервер NodeJS+Express та була розгорнута база даних PostgreSQL[17]. Amazon надає цілий рік безкоштовного користування за моделью IaaS, що дає змогу налаштувати

середовище власноруч під особисті потреби. PostgreSQL – потужна база даних, яка цілком відповідає потребам робочої моделі для зберігання показників з датчиків та інформацію про користувачів системи.

Схема з’єднання показана на рисунку 3.4[Рис. 3.4]

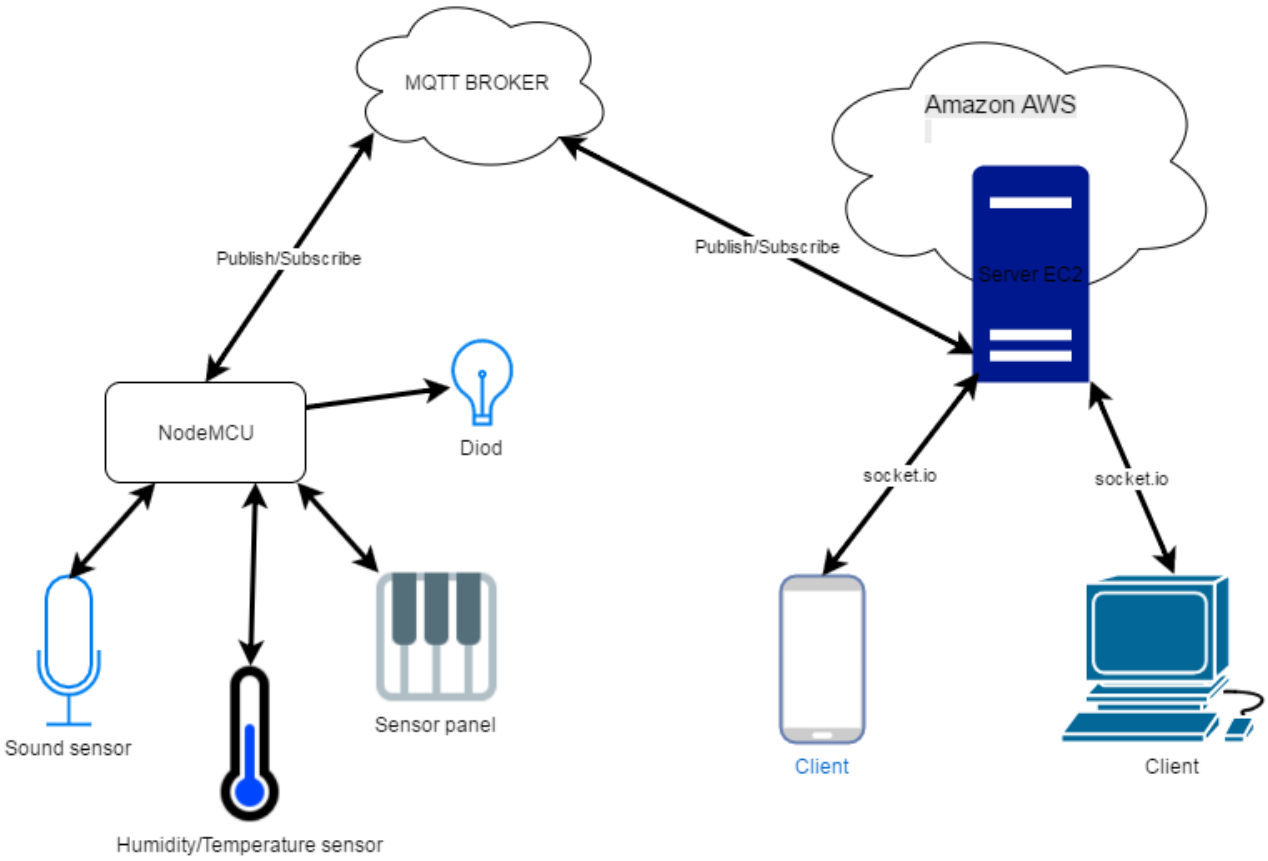

Рис. 3.4 – Схема зв'язку робочої моделі з хмарним керуванням

Був розроблений веб-інтерфейс[Рис. 3.5] для моніторингу температури та вологості з датчиків. Також є можливість спостерігати за звуковим шумом, який перевищує встановлений пороговий. Є можливість дистанційно ввімкнути або вимкнути світловий діод, який підключений до мікроконтролера.

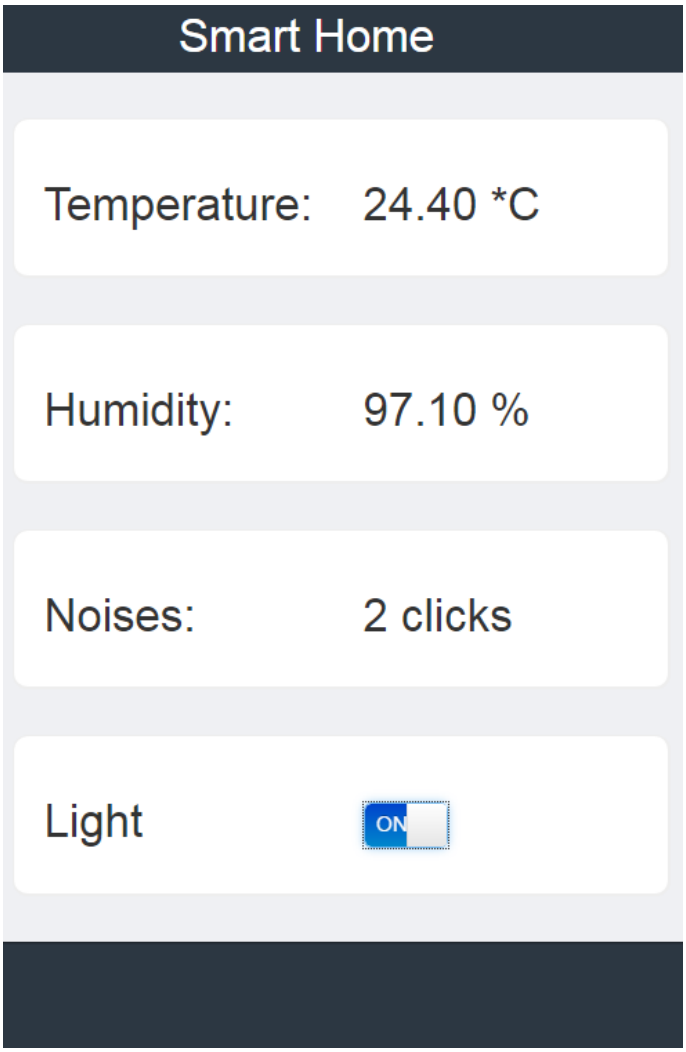

Рис 3.5 – Веб-інтерфейс моніторингу даних

Стартова сторінка сайту – вікно логіну[Рис 3.6]. Доступ до даних з датчиків, а також керування пристроями розумного стороннім особам не допускається. Тільки авторизовані користувачі мають доступ до будь-яких маніпуляцій та моніторингу

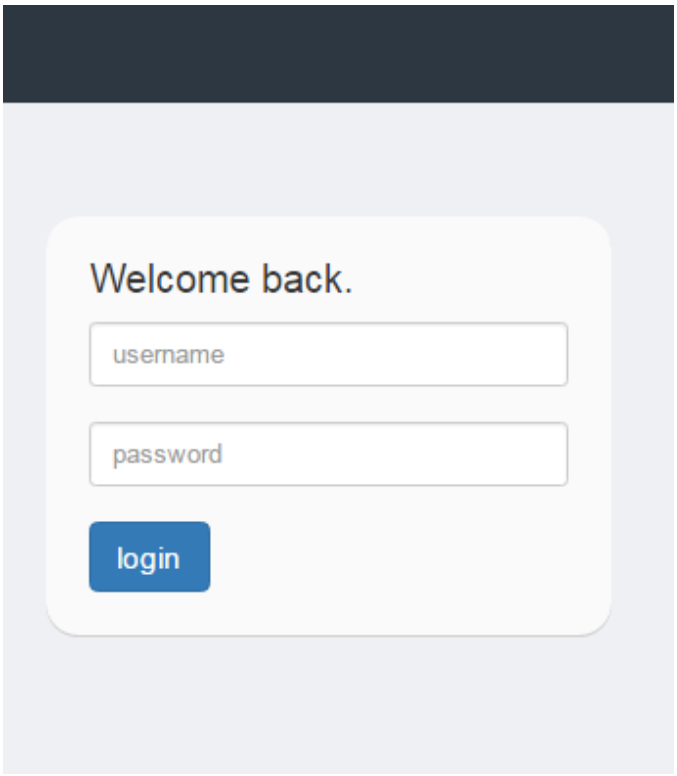

Рис. 3.6 – Стартове вікно логіну сайта

Коли дані з MQTT брокера приходят на сервер, то раз на десять показників вологості і температури, середнє значення посилається базу данних у відповідну таблицю з часовим штампом для можливості подальшого аналізу та побудови інфографічних моделей.

3.4 Висновок до розділу 3

Хоча тестова схема і відображає малий набір даних, проте з точки зору моделі, яку вона відображає – робоча модель має великий потенціал до розширення. Всі датчики є абстрактин уявленням більш складних приладів, які з легкістю можуть бути впроваджені в систему, розширюючи її до потреб користувача Були продемонстровані всі типи хмарного керування в обидва напрямки: датчики-сайт, сайт-датчики.

Також, важливим є той факт, що дана система хмарного керування розумним будинком (з невеликою кількістю змінень) може бути впроваджена в систему, з використанням технологій інших хмарних сервісів, адже робоча модель використовує загальноприйняті та актуальні засоби, такі як протокол MQTT та socket.io для передачі даних, мову програмування Javascript та мову розмітки HTML/CSS для візуалізації у веб-застосунках, базу даних PostgreSQL для збереження даних та мову програмування C для налаштування мікроконтролера.

4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для управління розумним домом за допомогою хмарних обчислень

Програмний продукт є кросплатформеним і може використовуватись, як на персональних комп'ютерах під управлінням операційної системи Windows, так і на ПЕОМ на яких встановленні UNIX-подібні операційні системи.

Нижче наведено аналіз різних варіантів реалізації ПО з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який буде встановлений у хмарний сервіс та клієнтська частина, з якою буде взаємодіяти користувач на ПК або Android пристрої. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування клієнтської частини;

F_2 – вибір мови програмування серверної частини;

F_3 – вибір API для хмарного керування розумним будинком.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Java;

б) мова програмування C++;

Функція F_2:

а) мова програмування Java;

б) мова програмування PHP;

Функція F_3:

а) API Arduino;

б) API onion.io.

4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

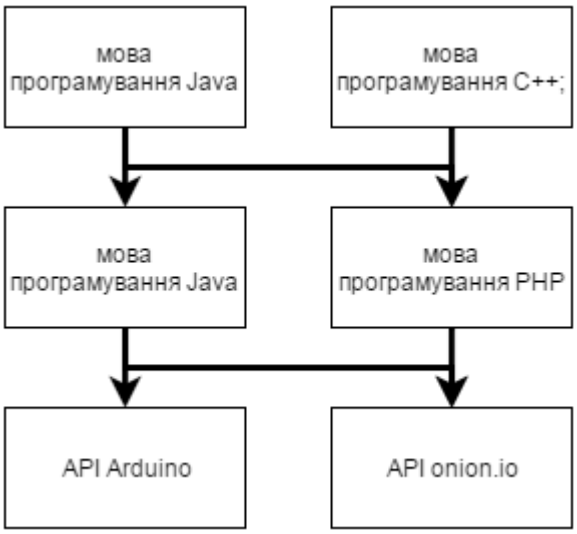

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Займає менше часу при написанні коду	Повільне виконання коду
	Б	Код швидко виконується, кросплатформений	Займає більше часу при написанні коду
F2	А	Займає менше часу при написанні коду	Повільне виконання коду
	Б	Код швидко виконується	Займає більше часу при написанні коду, не кросплатформений
F3	*А*	Ефективний код без надлишків	Складний у використанні
	Б	Простий у використанні	Малий перелік можливостей

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки написана програма на Java може бути легко влаштована як для Android пристроїв, так и і для ПК, то варіант з швидшою реалізацією є пріоритетнішим і варіант б) має бути відкинутий.

Функція *F2:*

Оскільки розрахунки проводяться з невеликими об’ємами вхідних даних, то час виконання програмного коду не є суттєвим показником, тому варіант з швидшою реалізацією є пріоритетнішим і варіант б) має бути відкинутий.

Функція *F3:*

Так як обидва варіанти API мають як сильні так і слабкі сторону, тому обидва варіанти а) і б) гідні розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1a – F2a – F3a
2. F1a – F2a – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.2 Обґрунтування системи параметрів ПП

4.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – час на вивчення API та написання коду;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час отримання данних з хмари;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає час, за який буде вивчена документація і протестована тестова схема

X2: Відображає об’єм пам’яті в хмарі, яка буде зберігати данні з датчиків розумного будинку

X3: Відображає час, за який запит в хмару повернеться до клієнта.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику

4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Час на вивчення API та написання коду	X1	год	10	6	2
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час отримання данних з хмари	X3	мс	1200	500	120
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

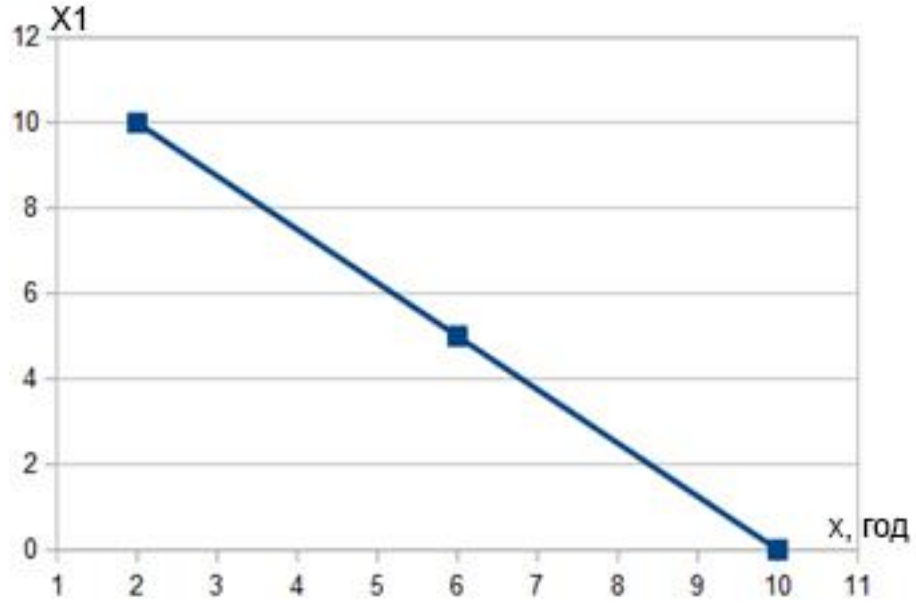

Рисунок 4.2 – X1, Час на вивчення API та написання коду

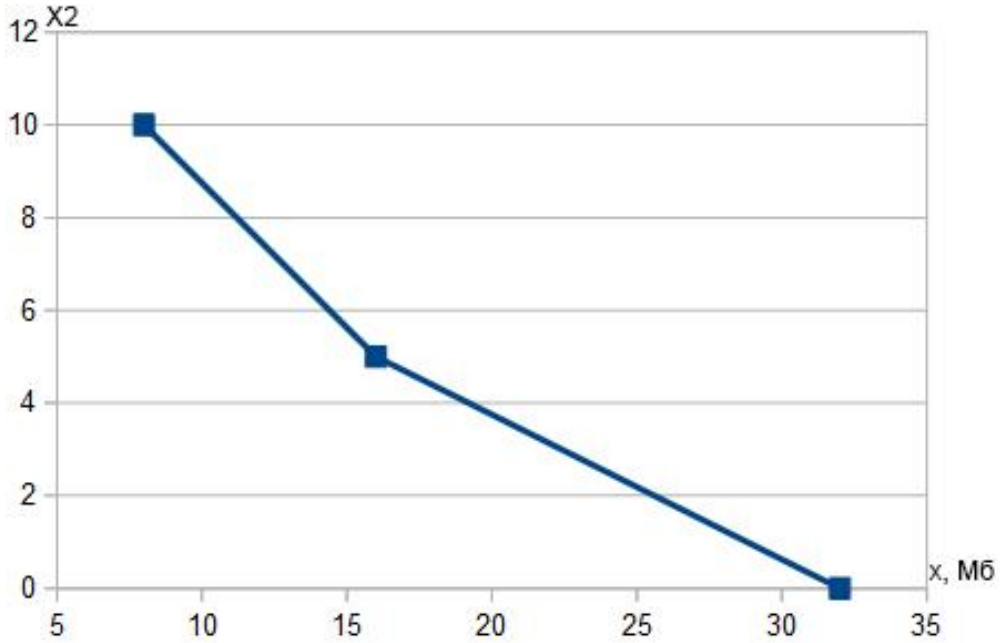

Рисунок 4.3 – X2, Об’єм пам’яті для збереження даних

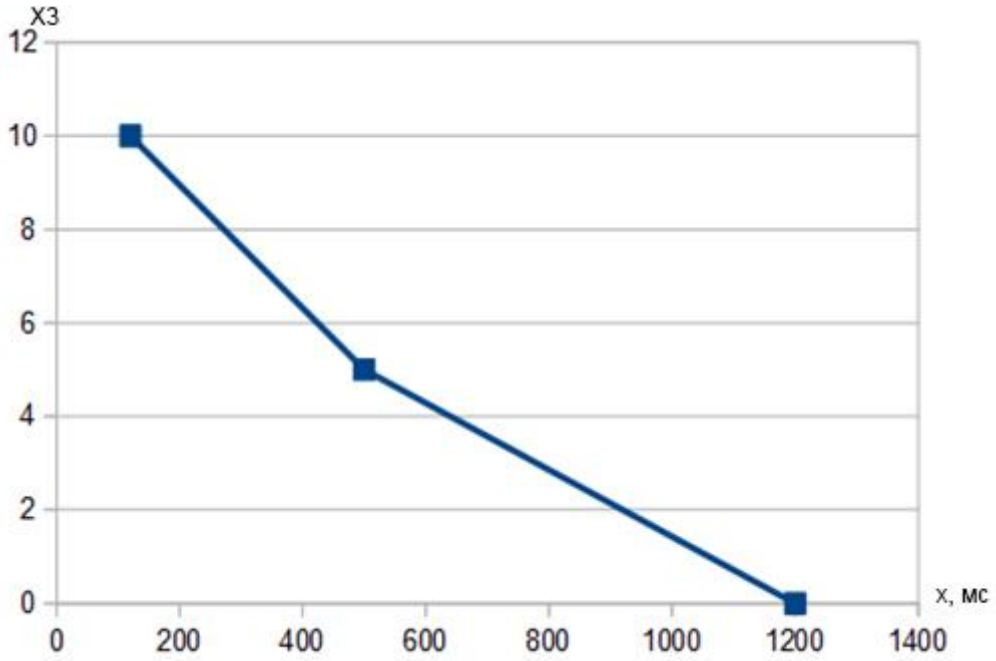

Рисунок 4.4 – X3, Час отримання данних з хмари

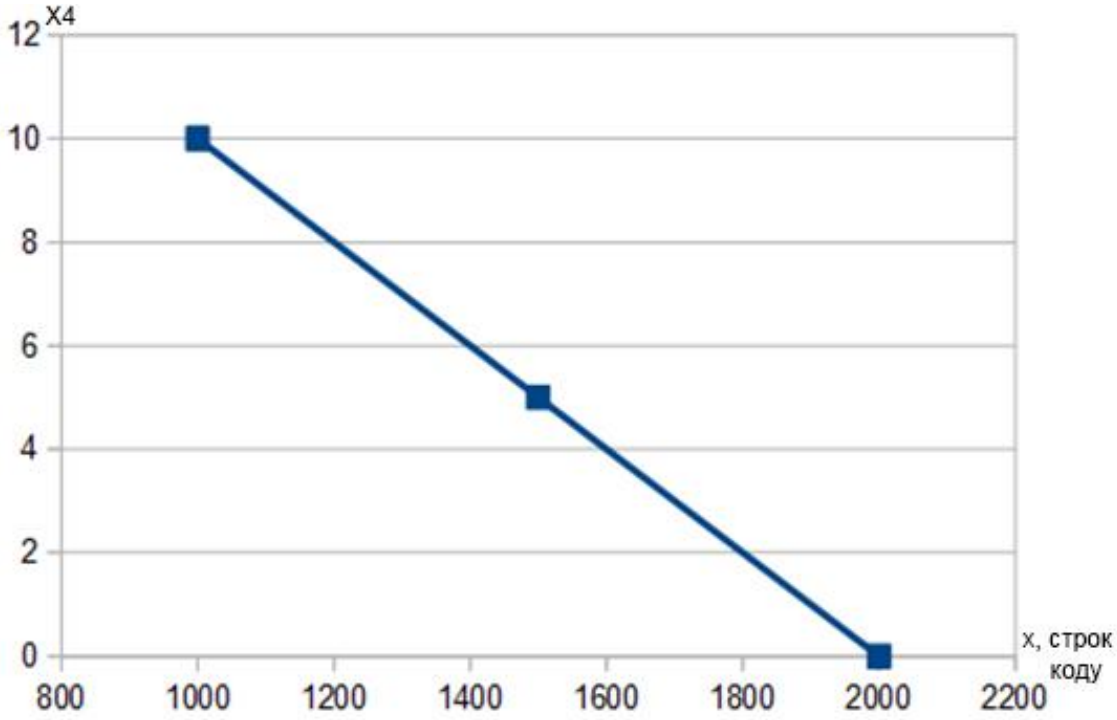

Рисунок 4.5 – X4, Потенційний об’єм програмного коду

4.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Час на вивчення API та написання коду	год	1	2	2	2	1	2	2	12	-5,5	30,25
X2	Об’єм пам’яті для збереження даних	Мб	2	1	1	1	2	1	1	9	-8,5	72,25
X3	Час отримання данних з хмари	мс	3	3	4	3	4	4	3	24	6,5	42,25
X4	Потенційний об’єм програмного коду	кількість строк коду	4	4	3	4	3	3	4	25	7,5	56,25
	Разом		10	10	10	10	10	10	10	70	0	201

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 201.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 201}{7^2(4^3 - 4)} = 0{,}82 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю4.4.

Таблиця4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	<	<	<	>	<	<	<	0,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	<	>	<	<	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij}визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}. \quad (4.1)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j. \quad (4.2)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	1,5	1,5	4,5	0,281	16,25	0,275	59,125	0,274
X2	1,5	1,0	1,5	1,5	5,5	0,344	21,25	0,360	77,875	0,361
X3	0,5	0,5	1,0	1,5	3,5	0,219	12,25	0,208	44,875	0,207
X4	0,5	0,5	0,5	1,0	2,5	0,156	9,25	0,157	34,125	0,158
Всього:					16	1	59	1	216	1

4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X1(Час на вивчення API та написання коду) та X4 (Потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X2(Об’єм пам’яті для збереження даних) буде найкращим у випадку обрання у F3 варіанта Б і становитиме 10, для варіанту А це значення буде 28.

Абсолютне значення параметра X3(Час отримання данних з хмари) буде найкраще при обрані варіанту А 180, а при обрані варіанту Б воно буде середнім 900.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} \,, \qquad (4.3)$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	3	9	0,361	3,249
F2	А	X4	1500	5	0,207	1,035
F3	А	X2	10	9	0,158	1,422
		X3	180	9	0,274	2,466
	Б	X2	28	2	0,158	0,316
		X3	900	2	0,274	0,548

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \quad (4.4)$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 3,249 + 1,035 + 1,422 + 2,466 = 8,172

K_{K2} = 3,249 + 1,035 + 0,316+ 0,548= 5,148

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.4 Економічний аналіз варіантів розробки ПП

Програмний продукт, що буде встановлюватись на сервер кафедри вже розроблено, проте необхідно увесь рік тримати під контролем сервер, на якому буде встановлено програмне забезпечення. Отже

T_1 = 365 людино-днів на рік.

$T_I = 365 \cdot 8 = 2920$ людино-годин;

Обидва варіанти однаково трудоємні

В адмініструванні беруть участь два адміністратори з окладом 5000 грн. Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.}, \quad (4.5)$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_Ч = \frac{5000 + 5000}{2 \cdot 21 \cdot 8} = 29{,}76 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot T_i \cdot К_Д, \quad (4.6)$$

де $С_ч$– величина погодинної оплати праці програміста;T_i – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$С_{ЗП}$ = 29,76 * 2920 · 1.2 = 104285,71 грн на рік.

Відрахування на єдиний соціальний внесок незалежно від групи професійного ризику становить 22%:

I. $С_{ВІД} = С_{ЗП} · 0.22$ =104285,71 * 0.22 =22,942,86 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Працюватиме одна електрона обчислювальна машина, котра працюватиме цілодобово:

$С_Г = 12·М·К_З$ = 12 · 5000· 0,2 = 12000 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г · (1+ К_З)$ = 12000· (1 + 0.2) = 14400 грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} · 0.22$ = 14400 * 0,22 = 3168 грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 25000 грн.

$С_А = К_{ТМ} · К_А·Ц_{ПР}$ = 1.15 · 0.25 · 25000 = 7187,5 грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ}·Ц_{ПР} · К_Р$ = 1.15 · 25000 · 0.05 = 1437,5 грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = Д_К \cdot t_З \cdot К_В = 365 \cdot 24 \cdot 1 = 8760$ годин,

де $Д_К$ – календарна кількість днів у році; t –кількість робочих годин електронної обчислювальної машини в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 8760 \cdot 0,6 \cdot 1 \cdot 2,0218 = 10626,58$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 25000 \cdot 0,67 = 16750$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \quad (4.7)$$

$С_{ЕКС} = 14400 + 3168 + 7187,5 + 1437,5 + 10626,58 + 16750 = 53569,58$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 53569,58 / 8760 = 6,12$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М\text{-}Г} \cdot Т$$

$C_М = 7,26 * 8760 = 53569,58$

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0,67$$

$C_Н = 104285,71 * 0,67 = 69871,43$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н \quad (4.8)$$

$C_{ПП} = 104285,71 + 22942,86 + 53569,58 + 69871,43 = 250669,58$ грн.;

4.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \quad (4.9)$$

$К_{ТЕР1} = 8,172 / 250669,58 = 3,26 \cdot 10^{-5}$;

$К_{ТЕР2} = 5,148 / 250669,58 = 1,81 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 3,26 \cdot 10^{-5}$.

4.6 Висновки до розділу 4

В даному розділі проведено повний функціонально-вартісний аналіз програмного забезпечення, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

У першій проведено дослідження програмний продукт з технічної точки зору: були поставлені основні параметри, що повинні бути головними при обранні кращої реалізації. На основі отриманих значень параметрів, оцінок експертів було обчислено коефіцієнт технічного рівня, який надалі у другій частині допоміг обрати найкращий варіант з техніко-економічної точки зору.

У другій частині виконувалося економічне обґрунтування альтернативних варіантів реалізації. Порівняння робились з урахуванням витрат на заробітні плати, електроенергії, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється. Після проведення першої частини аналізу було виявлено, що перший варіант є найбільш оптимальним для реалізації. Його показник техніко-економічного рівня якості $К_{ТЕР1}= 3{,}26 \cdot 10^{-5}$;

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування клієнтської частини – Java;
– мова програмування серверної частини – Java;
– вибір API для хмарного керування розумним будинком – Arduino.

Даний варіант виконання програмного комплексу є простим у написанні, достатньо швидким для користувача та гнучким у налаштованні та маштабуванні.

ВИСНОВКИ

За результатами аналізу історії розвитку та кінцевої стадії концепції розумного дома та методів передачі даних можно зробити висновок, що історія розвитку пройшла довгий та повний оновлень шлях, знайшовши найоптімальнішу концепцію роботи.

Були проаналізовані платформи хмарного керування розумним будинком, такі як: AWS IoT (Amazon), Google Cloud Platform IOT Solutions, Smart Home Cloud API (Samsung Smart Home) за чотирма параметрами:

1. Безпека передачі даних
2. Можливості розширення
3. Різноманітність пристроїв, які можуть бути підключені
4. Моніторинг та аналіз даних

В аспекті безпеки передачі даних всі три платформи виявилися однаково надійними, реалізуючи передові технології в шифруванні данних та аутентифікації користувачів та пристроїв.

В аспекті можливостей розширення домінує платформа Google IoT через її великий набір додаткових засобів для аналізу та моніторингу даних. Враховуючи те, що всі технології цієї платформи є платними і те, що можна легко знайти безкоштовні аналоги та впровадити їх до платформи Amazon IoT, то в даному аспекті немає чіткого лідера і все залежить від навичок розробника, часу та бюджету, які є в наявності у проекта. Samsung IoT в данному випадку подається з уже готовим набором інструментів, який важно кастомізувати під власні потреби.

В аспекті різноманітності пристроїв, які можуть бути підлючені Samsung IoT значно відстає через те, що він дозволяє підлючення тільки пристроїв, які були виготовлені компанією Samsung. На відміну від Amazon Iot та Google IoT,

які надають розробникам більшу свободу у виборі потрібних датчиків, пристроїв, протоколів та контролерів.

В аспекті моніторингу та аналізу даних всі платформи надають потрібний і достатній набір інструментів для адміністрування та налаштування приладів та датчиків. Google IoT має потужні інструменти візуалізації даних, які в свою чергу можна легко імплементувати до Amazon IoT. Samsung IoT лише дає опис своїх засобів без можливості протестувати їх на тестових даних, що є великим недоліком, коли мова йде про вибір платформи.

Хоча і розроблена тестова схема і відображає малий набір даних, проте з точки зору моделі, яку вона відображає – робоча модель має великий потенціал до розширення. Всі датчики є абстрактин уявленням більш складних приладів, які з легкістю можуть бути впроваджені в систему, розширюючи її до потреб користувача Були продемонстровані всі типи хмарного керування в обидва напрямки: датчики-сайт, сайт-датчики.

Також, важливим є той факт, що дана система хмарного керування розумним будинком (з невеликою кількістю змінень) може бути впроваджена в систему, з використанням технологій інших хмарних сервісів, адже робоча модель використовує загальноприйняті та актуальні засоби, такі як протокол MQTT та socket.io для передачі даних, мову програмування Javascript та мову розмітки HTML/CSS для візуалізації у веб-застосунках, базу даних PostgreSQL для збереження даних та мову програмування C для налаштування мікроконтролера.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється. Після проведення першої частини аналізу було виявлено, що варіант з мовою програмування сервеної та клієнтської частин на мові javascript, та використання хмарної платформи Amazon та їх послугу виділення віртуального серверу є найефективнішим. Також враховуючи те, що

автор дипломної роботи добре знайом з цими технологіями, то можно бути впевненими, що використовуються найкращі практики у створенні архітектури коду та його написання. Показник техніко-економічного рівня якості $К_{ТЕР1}$= $3{,}26 \cdot 10^{-5}$, що вілповідає передбаченням.

ПЕРЕЛІК ПОСИЛАНЬ

1. Черняк Л. Интернет вещей: новые вызовы и новые технологии // Открытые системы. — 2013. — № 04.
2. Интеграция и взаимодействие в сети Веб — Режим доступа: http://www.intuit.ru/studies/courses/485/341/lecture/8211. — Дата доступа: 06.06.2016р
3. Tesla Nikola. Method of and apparatus for controlling mechanism of moving vessels and vehicles//Patent 613809. — United States Patent and Trademark Office, 8 Листопада 1898
4. Spicer Dag If You Can't Stand the Coding, Stay Out of the Kitchen. — Dr. Dobb's Journal — Випуск 2 вересня 2010
5. Офіційний сайт компанії Amazon — Режим доступа: https://aws.amazon.com/ru/iot/how-it-works/ — Дата доступа: 06.06.2016р
6. Офіційний сайт компанії Google — Режим доступа: https://cloud.google.com/solutions/iot/ — Дата доступа: 06.06.2016р
7. Офіційний сайт компанії Samsung — Режим доступа: http://developer.samsung.com/smart-home — Дата доступа: 06.06.2016р
8. Офіційний сайт компанії Amazon — Режим доступа: https://aws.amazon.com/ru/pricing/?nc2=h_ql_ny_livestream_blu — Дата доступа: 06.06.2016р
9. Сайт MQTT — Режим доступа: http://mqtt.org/ — Дата доступа: 06.06.2016р
10. Офіційний сайт документації NodeMCU — Режим доступа: https://nodemcu.readthedocs.io — Дата доступа: 06.06.2016р
11. Офіційна сторінка NodeMCU на порталі GitHub — Режим доступа: https://github.com/nodemcu/nodemcu-firmware — Дата доступа: 06.06.2016р
12. Офіційний сайт сервісу cloudmqtt — Режим доступа: https://www.cloudmqtt.com/ — Дата доступа: 06.06.2016р

13. Сторінка бізнес-планів cloudmqtt — Режим доступа: https://www.cloudmqtt.com/plans.html — Дата доступа: 06.06.2016р
14. Офіційна сторінка NodeJS на російській мові — Режим доступа: http://nodejs.ru/ — Дата доступа: 06.06.2016р
15. Офіційна сторінка Express для NodeJs — Режим доступа: http://expressjs.com/uk/ — Дата доступа: 06.06.2016р
16. Офіційна сторінка socket.io — Режим доступа: http://socket.io/ — Дата доступа: 06.06.2016р
17. Офіційний сайт postgresql — Режим доступа: https://www.postgresql.org/ — Дата доступа: 06.06.2016р