Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Кібербезпеки_____
(повна назва)

Кафедра _____Інформаційно-мережної інженерії_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____
_____Проєктування інформаційно-керуючої системи житлового_____
_____середовища на базі IoT-технологій_____
(тема)

Виконав:
здобувач ___4_____ року навчання,
групи _____ТРІМІ-22-2_____
_____Данило СУХОТЕНКО_____
(власне ім'я, прізвище)

Спеціальність _____172 Телекомунікації
_та__радіотехніка_____
_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
Освітня програма ___Інформаційно-мережна
_інженерія_____
_____
(повна назва освітньої програми)

Керівник _доц. Галина ЛЯШЕНКО_____
(посада, власне ім'я, прізвище)

Допускається до захисту

В.о. завідувача кафедри ІМІ _____ Дарія ЧЕБОТАРЬОВА
(підпис) (власне ім'я, прізвище)

2026 р.

Не містить відомостей заборонених до відкритого публікування.

Студент / Данило СУХОТЕНКО/

Керівник / Галина ЛЯШЕНКО /

Харківський національний університет радіоелектроніки

Факультет Кібербезпеки

Кафедра Інформаційно-мережної інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 172 Телекомунікаціїї та радіотехніка
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

В.о. зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Сухотенко Данилу Владиславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Проєктування інформаційно-керуючої системи житлового середовища на базі IoT-технологій

затверджена наказом університету від 02 червня 20_26_ р. № _466Ст_

2. Термін подання здобувачем роботи до екзаменаційної комісії _13___ _липня__ 2026__ р.

3. Вихідні дані до роботи _ Обґрунтувати вибір бездротових протоколів IoT, розробити програмно-апаратну модель розумного будинку на базі мікроконтролера ESP32 із набором сенсорів та створити вебзастосунок для моніторингу телеметрії в режимі реального часу._

4. Перелік питань, що потрібно опрацювати в роботі

1. Теоретичні основи iot у системах автоматизації житла
2. Технологічна платформа для реалізації систем
3. Моделювання та налаштування системи
4. Програмна реалізація та вебінтерфейс системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
_Слайди презентації у Power Point _____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Ознайомлення із завданням. Уточнення ТЗ.	02.06.26 – 03.06.26	Виконано
2	Підбір літератури за темою роботи.	04.06.26 – 06.06.26	Виконано
3	Виконання розділу 1	08.06.26 – 11.06.26	Виконано
4	Виконання розділу 2	11.06.26 – 19.06.26	Виконано
5	Виконання розділу 3	20.06.26 – 30.06.26	Виконано
6	Виконання розділу 4	01.07.26 – 09.07.26	Виконано
7	Оформлення пояснювальної записки	10.07.26 – 13.07.26	Виконано
8	Оформлення презентаційного	14.07.26 – 16.07.26	Виконано
	матеріалу. Підготовка до захисту у ЕК		

Дата видачі завдання _02____ __червня_____2026__ р.

Здобувач _____
(підпис)

Керівник роботи _____ _доц. Галина ЛЯШЕНКО_____
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка має 67 сторінок, 18 рисунків, 3 таблиці, 14 джерел за переліком посилань.

РОЗУМНИЙ БУДИНОК, ІНФОРМАЦІЙНО-КЕРУЮЧА СИСТЕМА, MQTT-БРОКЕР, WI–FI, DHT22, MQ–2, PIR, МІКРОКОНТРОЛЕР ESP32

Метою кваліфікаційної роботи є створення архітектури та реалізація програмно-апаратної бездротової системи моніторингу для розумного будинку на основі IoT-технологій.

В ході виконання кваліфікаційної роботи досліджено бездротові протоколи IoT для впровадження концепції розумного будинку. Особливу увагу приділено зниженню затримок та використанню хмарних технологій для маршрутизації. Обґрунтовано переваги бездротового інтерфейсу Wi-Fi над дротовим рішеннями, це забезпечує гнучкість, високу енергоефективність та просте масштабування системи в житлі.

Сформульовано оптимальні інженерні рішення щодо побудови IoT-систем, здатних функціонувати в режимі реального часу. Розроблено функціональну програмно-апаратну модель розумного будинку на базі мікроконтролера ESP32 з вбудованим Wi-Fi модулем та набором сенсорів: цифровим датчиком температури DHT22, аналоговим датчиком MQ-2, а також інфрачервоним датчиком руху PIR. Програмне забезпечення мікроконтролера написане на мові програмування C++ з використанням протоколу MQTT для публікації даних. Для візуалізації та оперативного моніторингу стану об’єкта створено кросплатформений вебзастосунок на основі HTML, CSS, JavaScript, який забезпечує інтеграцію з хмарним MQTT-брокером та відображення телеметричних даних кінцевим користувачам.

ABSTARCT

The explanatory note contains 67 pages, 18 figures, 3 tables, 14 sources.

SMART HOME, INFORMATION AND MANAGEMENT SYSTEM, DATABASE, WI–FI, DHT22, MQ–2, PIR, ESP32 MICROCONTROLLER

The purpose of qualification work is to create an architecture and implement a software-hardware wireless monitoring system for a smart home based on IoT technologies.

During the qualification work, wireless IoT protocols were investigated for implementing the smart home concept. Special attention was paid to reducing delays and using cloud technologies for routing. The advantages of the Wi-Fi wireless interface over wired solutions were also substantiated, as it provides flexibility, high energy efficiency, and easy scaling of the system in housing.

Optimal engineering solutions have been formulated for building IoT systems capable of operating in real time. A functional hardware and software model of a smart home has been developed based on a high-perfomance ESP32 microcontroller with a built-in Wi-Fi module and a set of a sensors: a digital temperature and humidity sensor DHT22, an analog sensor MQ-2, and an infrared motion sensor PIR. The microcontroller software is written in C++ using the MQTT protocol for data publication. For visualization and operational monitoring of the object`s status, a cross-platform web application based on HTML, CSS, and JavaScript has been created, which provides integration with a cloud MQTT broker and display telemetry data to end users.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AIoT (Artificial Intelligence of Things) – концепція інтеграції штучного інтелекту з технологіями Інтернету речей.

API (Application Programming Interface) – програмний інтерфейс для взаємодії між програмними компонентами.

CSS (Cascading Style Sheets) – каскадні таблиці стилів для оформлення вебсторінок.

HTML (HyperText Markup Language) – стандартна мова розмітки вебсторінок.

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту.

IEEE (Institute of Electrical and Electronics Engineers) – міжнародна організація зі стандартизації в галузі електротехніки та електроніки.

IP (Internet Protocol) – міжмережевий протокол передачі даних.

IPv6 (Internet Protocol version 6) – шоста версія протоколу IP.

LPWAN (Low Power Wide Area Network) – енергоефективна глобальна мережа великого радіуса дії.

M2M (Machine-to-Machine) – технологія автоматичного обміну даними між пристроями.

MQTT (Message Queuing Telemetry Transport) – протокол обміну телеметричними повідомленнями між IoT-пристроями.

QoS (Quality of Service) – рівень якості доставки повідомлень у мережі.

SoA (Service-Oriented Architecture) – сервісно-орієнтована архітектура побудови інформаційних систем.

TCP/IP (Transmission Control Protocol / Internet Protocol) – стек мережевих протоколів для передавання даних.

UDP (User Datagram Protocol) – протокол передачі дейтаграм без встановлення з'єднання.

UUID (Universally Unique Identifier) – універсальний унікальний ідентифікатор.

ВСТУП

Швидкий розвиток технологій інтернету речей суттєво змінив підходи до автоматизації житлових просторів. Нині системи “розумного будинку” перестали бути розкішшю і стали необхідним засобом для забезпечення безпеки, комфорту та енергоефективності. Особливо важливим є доступні та гнучкі рішення на основі відкритих протоколів обміну даними та сучасних енергоефективних мікроконтролерів, що дозволяють створювати надійні системи моніторингу з миттєвим реагуванням на зміни в приміщенні та інформуванням користувача в режимі реального часу.

Традиційні системи автоматизації часто характеризуються високою вартістю, замкнутістю на конкретного виробника та складністю в експлуатації через необхідність серверів і баз даних. Альтернативним підходом є побудова розподілених систем моніторингу за принципом “Edge-to-Cloud”. Використання протоколу MQTT та хмарної інфраструктури дозволяє організувати безперервний потік телеметричних даних від датчиків до вебінтерфейсу користувача.

Метою роботи є розробка бездротової системи моніторингу типу “розумний будинок”, що поєднує програмну та апаратну складові й функціонує на базі мікроконтролера ESP32, забезпечуючи при цьому контроль параметрів мікроклімату, виявлення загрози пожежі та охорону приміщень у режимі реального часу. Для досягнення поставленої мети передбачається виконання таких завдань: здійснити моделювання взаємодії датчиків DHT22, MQ-2 та PIR у симуляційному середовищі Wokwi; налаштувати процес маршрутизації даних із застосуванням хмарного MQTT-брокера HiveMQ; а також створити кросплатформений вебзастосунок на основі технологій HTML, CSS та JavaScript, призначений для візуалізації отриманих показників.

1 ТЕОРЕТИЧНІ ОСНОВИ IoT У СИСТЕМАХ АВТОМАТИЗАЦІЇ

1.1 Концепція та історія розвитку Інтеренету речей

Суть концепції Інтернету речей полягає в трансформації звичайних фізичних об'єктів на активних учасників цифрового простору. На відміну від традиційного інтернету, орієнтованого на взаємодію між людьми, IoT створює глобальну мережу між самими предметами. Завдяки інтеграції мікроконтролерів, сенсорів та виконавчих механізмів, будь-який побутовий чи промисловий пристрій набуває здатності автономно збирати телеметрію, обмінюватися даними та взаємодіяти з навколишнім середовищем без прямого втручання людини.

Коли така інфраструктура розгортається в межах житлового простору, вона утворює систему “розумного будинку”. На сьогоднішній день IoT перетворився з перспективної ідеї на динамічну реальність, яка об'єднує датчики, контролери та програмне забезпечення в єдиний синхронізований організм. Ключовий алгоритм роботи системи базується на безперервному зборі локальних даних, їх передачі через мережеві протоколи до хмарних платформ або периферійних серверів, де інформація аналізується для автоматичного прийняття рішень.

У контексті розробленого проєкту така архітектура забезпечує надійний дистанційний моніторинг і контроль стану приміщення в режимі реального часу. Інтеграція фізичного простору з вебінтерфейсом користувача дозволяє не просто збирати статистику, а миттєво реагувати на критичні події. Завдяки безперервному аналізу інформації з датчиків, система здатна в автоматичному режимі ідентифікувати загрози безпеці, такі як поява диму, витік води чи фіксація стороннього руху, та оперативно надсилати сповіщення, мінімізуючи ризики для матеріальних ресурсів і мешканців.

Вихідною точкою впровадження технології IoT прийнято вважати 1982 рік, коли дослідники з Університету Карнегі-Меллона успішно модернізували

звичайний торговельний автомат із продажу напоїв Coca-Cola. Завдяки інтеграції апарату в локальну комп'ютерну мережу, його вдалося перетворити на перший в історії прототип побутового смарт-пристрою. Цей інноваційний на той час термінал набув здатності автономно відстежувати та дистанційно передавати інформацію щодо наявності товарних позицій усередині камери, а також контролювати поточні показники охолодження напоїв.

Подальший розвиток мережевих технологій вимагав теоретичного обґрунтування взаємодії фізичних об'єктів. Важливою етапом став 1990 рік, коли Марк Вейзер сформулював концепцію повсюдного комп’ютингу, яка передбачала інтеграцію обчислювальних потужностей у побутові предмети для автоматизації рутини [3]. Вже в 1993 році корпорація Microsoft реалізувала перший практичний досвід створення екосистеми M2M, презентувавши платформу “at Work”. Вона містила власну ОС та протокол для об'єднання офісної периферії, проте через неготовність тогочасної інфраструктури проєкт невдовзі згорнули [3].

Офіційне визначення напряму з'явилося у 1999 році, коли британський інженер Кевін Ештон запровадив термін “Інтернет речей” для опису системи, де датчики пов'язують фізичні об'єкти з віртуальним простором [2]. Державне визнання технологія отримала у листопаді 2008 року: Національна розвідувальна рада США включила IoT до звіту “Глобальні тенденції 2025” як перспективну розробку. Це вивело концепцію за межі суто наукових дискусій і перетворило її на важливу частину світової цифрової стратегії.

Перехід до масової комерціалізації та глобального масштабування інфраструктури відбувся в середині 2010-х років. Близько 2015 року стрибок у розвитку хмарних технологій зокрема, платформ AWS IoT та Microsoft Azure IoT забезпечив можливість агрегації та аналізу великих масивів даних. На ринку з'явилися перші стандартизовані споживчі екосистеми - Apple HomeKit, Google Home, а поява доступних та функціональних мікроконтролерів, наприклад, сімейства ESP32, зробила розробку IoT-пристроїв масовою, трансформувавши поодинокі смарт-гаджети у комплексні домашні мережі [2].

Подальше розгортання технологій на початку 2020-х років принесло зміну парадигми від простого збору інформації до її інтелектуального аналізу. У 2020 році чітко сформувався напрям AIoT, де алгоритми машинного навчання почали інтегруватися безпосередньо в автоматизацію. Водночас індустрія зіткнулася з гострою кризою фрагментації ринку: пристрої різних брендів були несумісними між собою, що змусило провідних технологічних гігантів розпочати спільну роботу над єдиним універсальним стандартом зв'язку.

На сьогоднішній день ключовим стандартом індустрії став універсальний протокол сумісності Matter, який остаточно подолав проблему фрагментації, дозволивши обладнанню від Apple, Google, Amazon та тисяч інших виробників безперешкодно взаємодіяти локально. Сучасна архітектура систем розумного будинку базується на периферійних обчисленнях, що гарантує високу швидкість обробки даних без обов'язкового надсилання приватного трафіку в хмару, а також на передових методах кіберзахисту кінцевих вузлів мережі. Завдяки такому послідовному розвитку, інтелектуальні системи пройшли еволюційний шлях від локального моніторингу окремого автомата до глобальних, безпечних та повністю автономних екосистем.

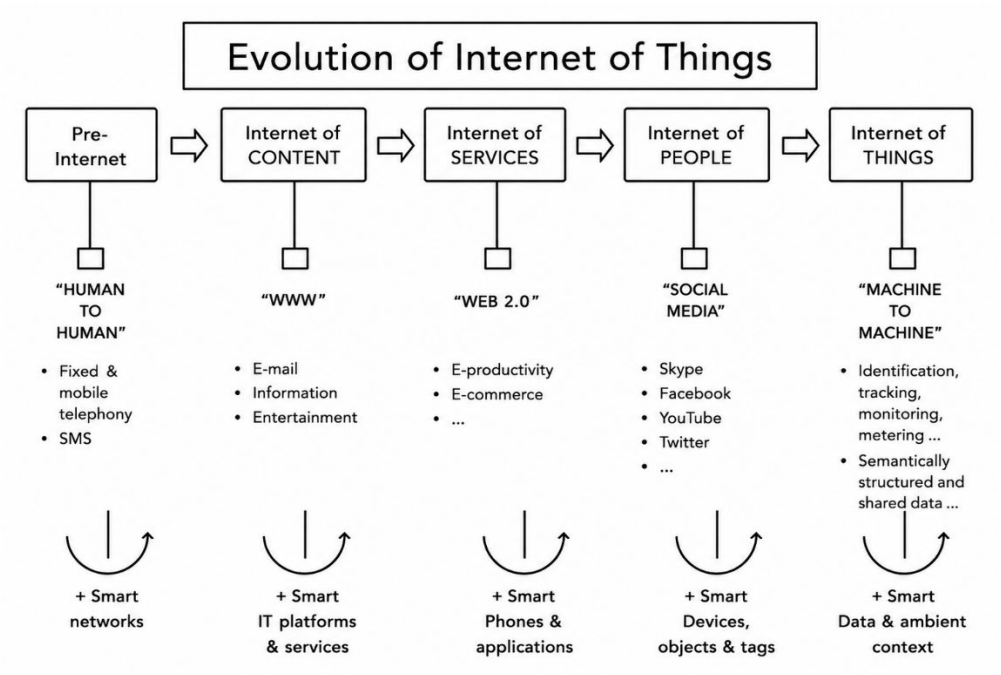

Рисунок 1.1 - Еволюція Інтернету речей

Спираючись на рис. 1.1 можна зазначити що перед тим, що є сьогодні, глобальна павутина пройшла п’ять основних історичних етапів:

- інтернет документів – створення перших електронних сторінок та бібліотек;
- інтернет контенту, тобто комерції – поява онлайн-банкінгу та перших торгових майданчиків;
- інтернет застосунків – ера технологій Web 2.0 та інтерактивних веб сайтів;
- інтернет людей – поширення у маси та розквіт соціальних мереж;
- інтернет речей – сучасний етап, де повноцінними учасниками обміну стають машини та пристрої.

Важливою архітектурною перевагою сучасних систем IoT є їхня здатність функціонувати в повністю автономному режимі, мінімізуючи безпосереднє втручання людини в процеси прийняття рішень та управління. На сьогоднішній день технологічні рішення на базі Інтернету речей демонструють високу ефективність у багатьох критично важливих галузях, таких як телемедицина та моніторинг стану пацієнтів, інтелектуальні транспортні системи та безпілотна логістика. Водночас у побутовому сегменті технології IoT сформували надійний технологічний фундамент для побудови комплексів безпеки, відеоспостереження, а також адаптивних систем автоматизованого кліматконтролю та енергоменеджменту.

Еволюція Інтернету речей продовжується і в теперішній час. Сучасні тенденції розвитку IoT тісно пов'язані з якісним покращенням засобів бездротового зв'язку, впровадженням високошвидкісних протоколів передачі даних із низькою затримкою, зокрема, мереж 5G/6G, технологій Wi-Fi 7 та енергоефективних стандартів LPWAN, а також модернізацією прикладних інтерфейсів. Важливим вектором сучасної модернізації галузі стала інтеграція штучного інтелекту в архітектуру інфраструктури AIoT. Це дозволяє реалізувати периферійні обчислення на кінцевих пристроях, забезпечуючи високу швидкість локальної обробки даних, покращення конфіденційності та кібербезпеки всієї системи.

Загальну структуру взаємодії кінцевих пристроїв, проміжних шлюзів та глобальних хмарних ресурсів представлено на рис. 1.2

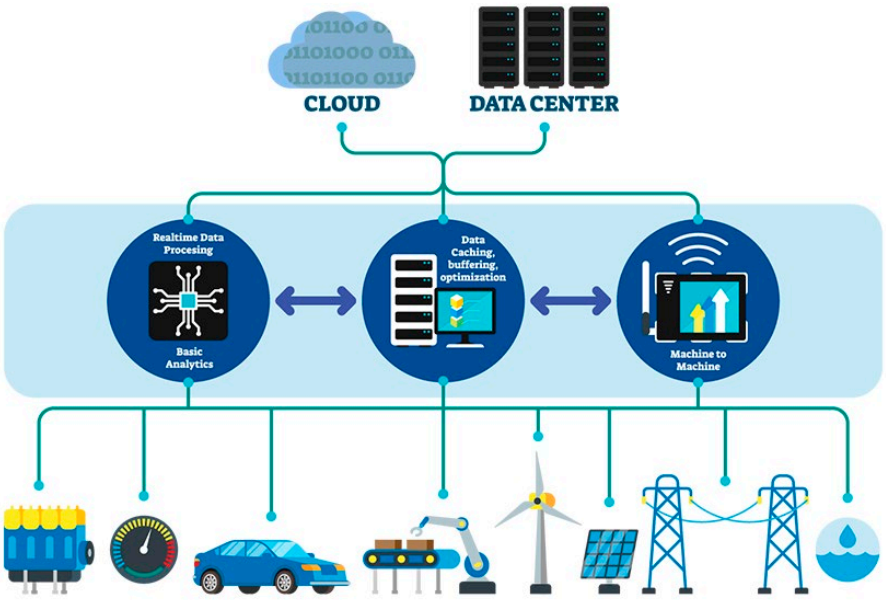

Рисунок 1.2 – Схема розподілу потоків даних в IoT

Можна сказати що сучасна екосистема Інтернету речей відійшла від схеми передачі даних “датчик-інтернет”, суть в тому, що між звичайними сенсорами та далекими від них хмарними серверами тепер стоїть помічник – мережевий шлюз.

1.2 Архітектура та функції рівнів IoT

Треба підкреслити, що однією з найголовніших інженерних вимог до Інтернету речей є побудова безперебійного зв’язку між усіма елементами мережі. Архітектурна модель системи повинна гарантувати високу стабільність, забезпечуючи надійну взаємодію фізичного простору кімнати з віртуальним цифровим середовищем. Проєктування такої системи є складним завданням, що охоплює бездротові технології, комунікаційні інтерфейси, методи первинної обробки телеметрії та механізми захисту даних.

Важливо враховувати такі характеристики, як розширюваність, масштабованість і ефективність обміну даними. Оскільки фізичні об’єкти можуть змінювати стан, переміщуватися або працювати в умовах реального часу, система має бути гнучкою та адаптивною. Сенсорні вузли повинні підтримувати безперервну передачу повідомлень навіть при зміні мережевих параметрів. Сучасний IoT має децентралізовану та гетерогенну структуру, що дозволяє об’єднувати різне обладнання в єдиний інформаційний комплекс [1].

Концепція SoA є однією з найперспективніших моделей для IoT-рішень. Ідея полягає у представленні складної системи як сукупності простих і чітко визначених сервісів або підсистем. Кожен програмний чи апаратний модуль може оновлюватися, обслуговуватися чи повторно використовуватися незалежно від інших частин системи. Такий підхід значно спрощує налаштування та розширення системи. Класична SoA-модель складається з функціональних рівнів (рис. 1.3 демонструє їх взаємодію).

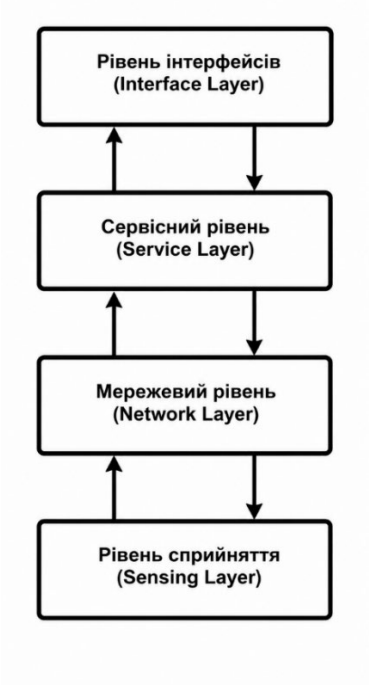

Рисунок 1.3 – Чотирирівнева сервісно-орієнтована архітектура

Рівень сприйняття (Sensing Layer), будь яка IoT-система починається з взаємодії з оточенням. Сучасний Інтернет речей – це глобальна мережа, де кожен

пристрій постійно перебуває на зв’язку. На рівні сприйняття інтелектуальні датчики, бездротові вузли та сенсори автоматично фіксують зміни фізичних параметрів довкілля [4].

Для точної ідентифікації в цифровому середовищі кожному пристрою присвоюється унікальний код, так званий UUID, який являє собою 128-бітове число. При розробці цього рівня системи необхідно враховувати низку принципових вимог:

- вартість та енергоспоживання сенсорних вузлів мають бути мінімізовані, щоб не створювати надмірного навантаження на систему живлення;
- знімання показників повинно відбуватися безперервно, без втрати пакетів даних;
- розміщення датчиків залежить від архітектурних особливостей приміщення і може бути як стаціонарним, так і довільним;
- топологія об'єднання пристроїв може набувати форми сітчастої, багатострибкової або локальної зіркоподібної структури.

Наступним етапом є своєчасна та захищена передача зібраних даних до пункту обробки. Цю функцію виконує мережевий рівень, який забезпечує базову взаємодію між усіма елементами екосистеми. Надійність мережевої інфраструктури тут набуває вирішального значення: система повинна автоматично виявляти щойно підключені мікроконтролери, присвоювати їм мережеві адреси та розподіляти функціональні ролі. При проєктуванні цього рівня основна увага приділяється вибору технології бездротового зв'язку.

Над мережевим рівнем розташовується сервісний рівень, який виконує роль центральної сполучної ланки всієї архітектури. Його доцільно розглядати як економічно ефективну віртуальну платформу, що дає змогу абстрагуватися від конкретної апаратної реалізації та зосередитися на роботі з програмними сервісами. Саме на цьому рівні реалізується ядро обробки інформації, здійснюється динамічний пошук активних пристроїв, встановлюються логічні зв'язки між ними, а також надаються стандартизовані програмні інтерфейси API.

Цей рівень організовано на базі хмарного MQTT-брокера HiveMQ. Брокер приймає потокову телеметрію від мікроконтролера ESP32 та одразу перенаправляє її підписникам за технологією “Edge-to-Cloud”.

Останнім, верхнім етапом архітектури є “Рівень інтерфейсів”, який відповідає за взаємодію системи з кінцевим користувачем або сторонніми програмами. Цей рівень вирішує проблему сумісності, він приводить різні потоки даних до єдиного стандартизованого вигляду.

Він реалізується перш за все у вигляді графічного вікна користувача, тобто Frontend, та програмних інструментів API. В проєкті цей рівень розроблено через HTML-структуру, візуальне оформлення та графічні елементи реалізовані в CSS, а динамічне відображення MQTT-повідомлень виконується совою JavaScript.

Можна зробити короткий висновок, що сучасні системи автоматизації все більш частіше використовують глибокі інтегровані стеки, замість спрощених варіантів.

1.3 Мережеві протоколи в сенсорних мережах

У сучасних інформаційно-керуючих системах житлового середовища до мережевих протоколів висувається низка вимог. Вони повинні забезпечувати надійну передачу даних, мінімальне енергоспоживання, можливість підключення великої кількості пристроїв, підтримку масштабування мережі, високу швидкість реагування та належний рівень інформаційної безпеки. Оскільки більшість IoT-пристроїв працює автономно від акумуляторів або батарей протягом тривалого часу, особливу увагу приділяють енергоефективним протоколам зв'язку.

Обмін даними між рівнями реалізується за допомогою різноманітних мережевих технологій, кожна з яких вирізняється своїми характерними перевагами та оптимальною сферою застосування.

Стандарт Wi-Fi базується на сімействі IEEE 802.11 та працює переважно в діапазонах 2,4 ГГц, 5 ГГц і 6 ГГц. Залежно від версії стандарту швидкість передавання даних може перевищувати 1 Гбіт/с, що дозволяє використовувати

його для камер відеоспостереження, мультимедійних систем, шлюзів IoT та іншого обладнання, яке передає великі обсяги інформації. Основним недоліком Wi-Fi є високе енергоспоживання, тому ця технологія рідко використовується в автономних сенсорних вузлах [5].

Для пристроїв із тривалим автономним живленням широко застосовується Bluetooth Low Energy, який входить до специфікації Bluetooth 5.x. Технологія забезпечує швидкість до 2 Мбіт/с та дальність зв'язку до 100–200 м у режимі Long Range. BLE використовується у фітнес-браслетах, медичних сенсорах, системах моніторингу та окремих компонентах розумного будинку.

Одним із найпоширеніших рішень для побудови сенсорних мереж є Zigbee, який базується на стандарті IEEE 802.15.4. Протокол працює переважно в діапазоні 2,4 ГГц, підтримує швидкість передачі до 250 кбіт/с та дозволяє об'єднати в одній мережі до 65 тисяч пристроїв. Використання топології Mesh забезпечує автоматичну маршрутизацію даних між вузлами, підвищуючи надійність зв'язку та розширюючи зону покриття без збільшення потужності передавачів.

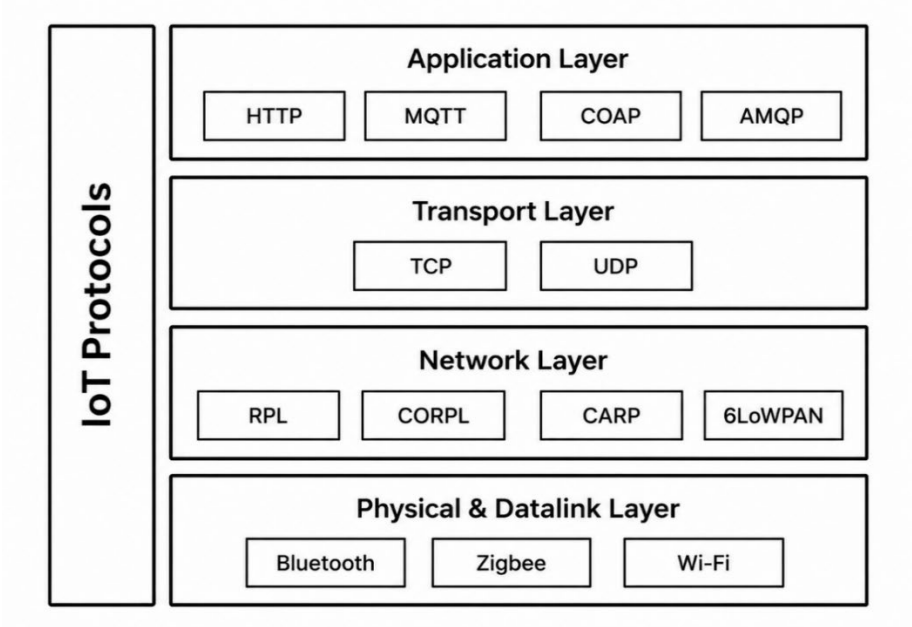

Рисунок 1.4 – Архітектура мережевої взаємодії в інфраструктурі IoT

Подібний принцип роботи використовує Thread, який також побудований на базі IEEE 802.15.4, але підтримує протокол IPv6 завдяки технології 6LoWPAN. Thread підтримує самоорганізовану Mesh-топологію, автоматичне відновлення маршрутів після виходу окремих вузлів із ладу та не має єдиної точки відмови, що підвищує стійкість усієї мережі

Сучасним стандартом взаємодії IoT-пристроїв є Matter, представлений організацією Connectivity Standards Alliance у 2022 році. Він працює поверх протоколу IP та підтримує Ethernet, Wi-Fi і Thread. Основною перевагою Matter є забезпечення сумісності пристроїв різних виробників без необхідності використання окремих екосистем або програмного забезпечення, що значно спрощує інтеграцію обладнання у системах розумного будинку.

Таблиця 1.1 – Порівняння базових протоколів маршрутизації в сенсорних мережах

Протокол	Дальність	Енергоспоживання	Швидкість
Wi-Fi (IEEE 802.11)	20–100 м	Високе	До 1 Гбіт/с і більше
Bluetooth Low Energy (BLE)	10–100 м	Дуже низьке	До 2 Мбіт/с
Zigbee (IEEE 802.15.4)	10–100 м	Низьке	До 250 кбіт/с
Thread (IEEE 802.15.4, IPv6)	10–100 м	Низьке	До 250 кбіт/с
Matter	Залежить від Wi-Fi або Thread	Низьке–середнє	Залежить від мережі
LoRaWAN	2–15 км	Дуже низьке	0,3–50 кбіт/с
MQTT	Залежить від мережі	Низьке	Залежить від транспортного протоколу

LoRaWAN працює в неліцензованих діапазонах 433 МГц, 868 МГц або 915 МГц, забезпечуючи передачу даних на відстань 2-5 км у міській забудові та до 15 км на відкритій місцевості. Швидкість передачі становить від 0,3 до 50 кбіт/с, що є достатнім для передавання телеметричної інформації при дуже низькому енергоспоживанні. Завдяки цьому автономні датчики можуть працювати від батареї протягом 5–10 років [14].

У IoT-системах використовуються спеціалізовані протоколи прикладного рівня. Найбільш поширеним є MQTT, який працює поверх TCP/IP за архітектурою Publisher–Broker–Subscriber. Протокол підтримує три рівні якості доставки повідомлень, що дозволяє обирати баланс між швидкістю передачі та надійністю доставки. Завдяки невеликому обсягу службових даних MQTT став фактичним стандартом обміну телеметрією між IoT-пристроями та хмарними платформами [14].

1.4 Огляд сучасних екосистем та платформ розумного будинку

На сьогоднішній день ідея “розумного будинку” повністю пройшла шлях від перших громіздких комп’ютерів типу ECHO IV до готових комерційних рішень, які може купити кожен. Сучасний етап розвитку автоматизації житла характеризується появою комплексних екосистем від великих світових гігантів IT-індустрії. Їх головна мета полягає в тому щоб об’єднати безліч різних датчиків та побутових приладів в одну просту та зрозумілу мережу [6].

Проте на практиці користувачі часто стикаються з проблемою несумісності, коли пристрої різних виробників не можуть безпосередньо взаємодіяти між собою. Для розв'язання цієї проблеми, компанії створюють універсальні програмні платформи, які виконують роль централізованого керування [6]. Вони дають змогу налаштовувати автоматичні сценарії та керувати технікою через один додаток або голосові команди. Щоб краще зрозуміти ринок IoT-технологій буде доцільно розглянути та порівняти найпопулярніші сучасні екосистеми.

Для аналізу було обрано наступні системи розумного будинку:

– Apple HomeKit;
– Xiaomi Mi Home.

1.4.1 Apple HomeKit

Платформа Apple HomeKit (пізніше стала Apple Home). Корпорація Apple вперше представила цей програмний фреймворк у вересні 2014 року разом з виходом iOS 8. Головною ідеєю створення єдиного інструменту для розробників, що дозволяє користувачам iPhone безпечно керувати розумними пристроями через голосового помічника Siri. У 2016 з'явився додаток "Home app" для зручного керування технікою [7].

Компанія Apple обрала унікальний підхід до побудови розумного будинку, який був та є орієнтованим на максимальну приватність та локальну обробку команд. Для фізичного зв'язку між пристроями використовується Wi-Fi та BLE разом із сучасним енергоефективним протоколом Thread. Для віддаленого доступу використовується домашній хаб, Apple TV або колонка HomePad, що є шифрованим мостом між домом і хмарою iCloud [7].

Рисунок 1.5 – Розумна колонка HomePad

Apple використовує закритий внутрішній протокол HAP, він працює поверх транспортних рівнів IP або BLE та вимагає обов’язкового шифрування за допомогою алгоритмів ChaCha20-Poly1305, а також автентифікації за технологією Ed25519. Щоб сторонні датчики та плати могли працювати в цій мережі, їх виробники мають пройти сертифікацію “Made for iPhone” та інтегрувати співпроцесори безпеки Apple.

1.4.2 Xiaomi Mi Home

Наступна платформа – це екосистема Xiaomi Mi Home (Mija). Вона є поширеною в Україні. Компанія стартувала розвивати цей напрямок у 2014 році. На відміну від Apple, Xiaomi створила широку мережу брендів разом із партнерами, такі як Aqara і Yeelight, які виробляютьтисячі різноманітних пристроїв, від датчиків температури до роботів-пилососів, які керуються через єдиний додаток Mi Home [8].

Рисунок 1.6 – Xiaomi Mi Home

Архітектура Xiaomi значною мірою залежить від хмари, більшість сценаріїв обробляються на віддалених серверах у Китаї або Європі. Для зв’язку система використовує три бездротові технології – Wi-Fi, ZigBee та Bluetooth.

Для обміну даними по Wi-Fi Xiaomi застосовує закритий протокол milO на основі UDP-пакетів із захистом через шифрування AES-128-CBC [8]. Основною проблемою системи є сильна залежність від закордонних серверів.

1.5 Висновки розділу

У розділі розглянуто історію Інтернету речей та протоколи сенсорних мереж. Продемонстровано, що великі системи вимагають складних алгоритмів для ефективного збереження енергії пристроїв.

Також проведено аналіз популярних Apple HomeKit і Xiaomi Mi Home з використовуваними ними технологіями, виявлено основні недоліки, закритість коду та залежність від сторонніх серверів.

2 ТЕХНОЛОГІЧНА ПЛАТФОРМА ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Мікроконтролер ESP32 з Wi-Fi

Інформаційно-керуюча система розумного будинку об'єднує в єдину мережу різноманітні сенсори, виконавчі механізми, контролери та засоби зв'язку, що забезпечують автоматизоване керування інженерними системами будівлі. Для стабільної та ефективної роботи такої системи необхідно правильно підібрати апаратні компоненти, які відповідають вимогам щодо продуктивності, енергоефективності, надійності та сумісності між собою. Не менш важливим є використання сучасних мережевих протоколів, які забезпечують безпечний і безперервний обмін даними між усіма елементами системи та підтримують можливість її подальшого розширення.

Для створеної системи було обрано мікроконтролер ESP32, датчик температури DHT22, датчик задимленості MQ-2, датчик руху PIR (HC-SR501).

Основним елементом апаратної частини є мікроконтролер ESP32. Це дуже популярна маленька плата, яка є по-перше недорогою, а по-друге має великі технічні можливості. Її головна унікальність полягає в тому, що на одному крихітному чіпі виробники вмістили не лише процесор для обчислень, а й повноцінні модулі для бездротового зв’язку – Wi-Fi та Bluetooth [9]. Це означає, що пристрою не потрібні кабелі з високою вартістю. чи зовнішні мережеві плати, щоб виходити в інтернет.

Всередині цієї плати працює двоядерний процесор із тактовою частотою до 240 МГц. Його ядра створені на базі 32-бітної технології Tensilica Xtensa LX6. Розумний будинок повинен робити кілька справ одночасно, а одне ядро встановлене на старих платах часто працює з помилками.

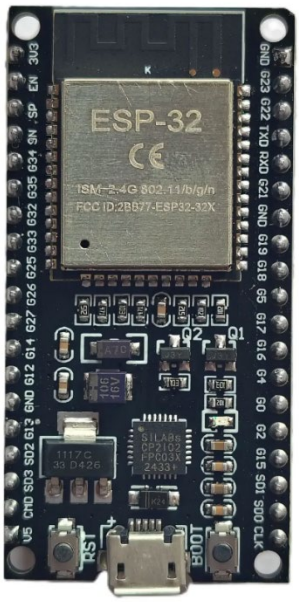

Рисунок 2.1 – Мікроконтролер ESP32

Для швидкої роботи програм плата оснащена 520 КБ оперативної пам’яті, 448 Кб постійної пам’яті для завантажувача, а також 16 Кб спеціальної енергонезалежної пам’яті RTC SRAM, яка зберігає критичні дані навіть у режимі глибокого сну.

Вбудований Wi-Fi модуль працює за стандартами 802.11 b/g/n і підтримує швидкість передачі даних до 150 Мбіт/с, чого з запасом вистачає для мереж IoT. Разом з цим плата сумісна з класичним Bluetooth v4.2 та енергоефективним BLE для швидкого з’єднання з іншими пристроями [10]. Також мікроконтролер має до 34 програмованих контактів GPIO, у тому числі в ньому є доступ до 2 каналів 8-бітного цифро-аналогового перетворювача.

Таблиця 2.1 – Технічні характеристики мікроконтролеру ESP32

Тип	Параметр
1	2
Заводський час	4 тижні
Монтажний тип	Поверхневе кріплення
Пакет / кейс	Модуль

Продовження таблиці 2.1

1	2
Робоча температура	-40 °C ~ 85 °C
Статус частини	Активний
Рівень чутливості до вологи (MSL)	Не застосовується
Напруга - постачання	2,7 В ~ 3,6 В
Частота	2,4 ГГц ~ 2,5 ГГц
Швидкість передачі даних	150 Мбіт/с
Протокол	802.11b/g/n, Bluetooth v4.2 + EDR, клас 1, 2 і 3
Потужність - вихід	20,5 дБ
Сім'я/стандарт RF	Bluetooth, Wi-Fi
Чутливість	-97 dBm
Серійні інтерфейси	ADC, GPIO, I2C, I2S, PWM, SDIO, SPI, UART
Модуляція	CCK, DSSS, OFDM
Статус ROHS	ROHS сумісна

2.2 Інфраструктура на базі MQTT-брокера HiveMQ

Аби досягти оперативного обміну повідомленнями між платою ESP32 та клієнтською частиною вебінтерфейсу, у межах проєкту було реалізовано мережеву інфраструктуру, побудовану на основі протоколу MQTT. Це відкритий легкий прикладний протокол типу Push, розроблений для надійної передачі даних у середовищах з обмеженою пропускною здатністю та обмеженими ресурсами мікроконтролерів.

Оскільки система працює в режимі реального часу без використання проміжних баз даних, MQTT є оптимальним рішенням завдяки компактності та мінімальним витратам на формування мережевих пакетів. Кожне повідомлення протоколу містить три компоненти: обов’язковий фіксований заголовок, необов’язковий змінний заголовок та безпосередньо корисне навантаження з даними сенсорів [11].

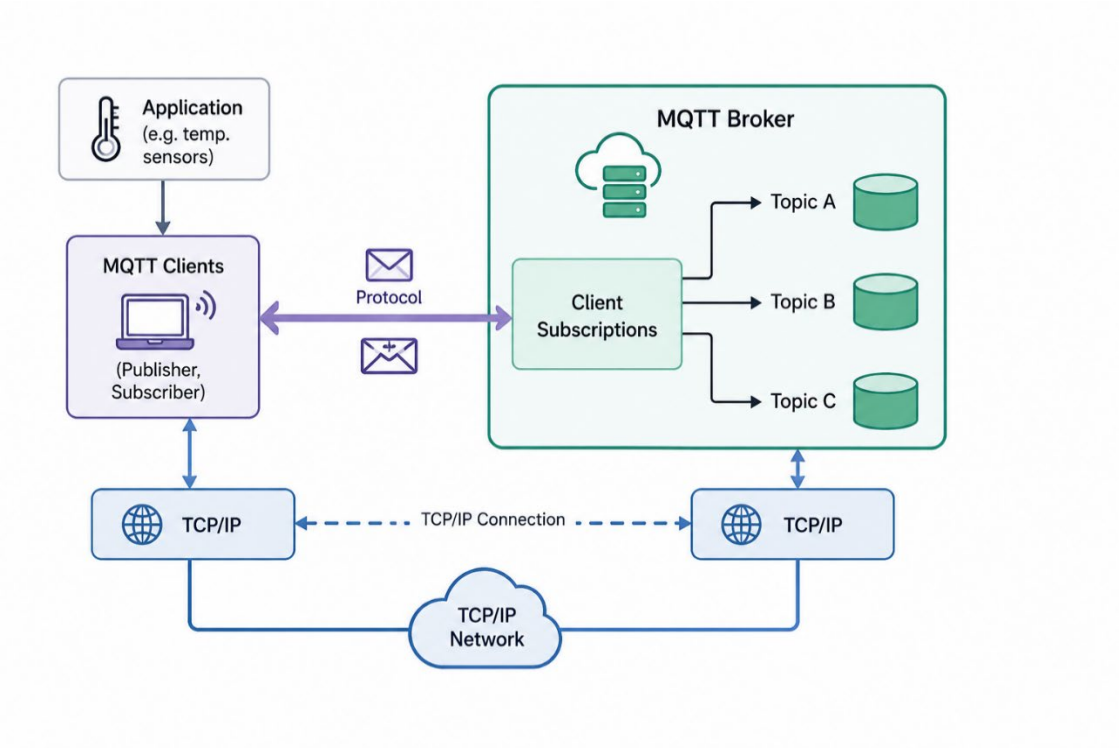

Рисунок 2.2 – Загальна архітектура взаємодії за протоколом MQTT

Архітектура зв’язку базується на моделі Publish/Subscribe. Клієнти не обмінюються даними безпосередньо, натомість цьому центральну роль координатора та маршрутизатора трафіку виконує хмарний MQTT-брокер HiveMQ. Інформація організується за допомогою текстових каналів “топіків”, які виступають темами для повідомлень.

Взаємодія елементів створеної системи відбувається наступним чином:

1) Publish

Мікроконтролер ESP32 виступає видавцем, зчитуючи показники з датчиків руху PIR, диму MQ-2 та мікроклімату DHT22, пакує їх у текстовий формат і надсилає брокеру HiveMQ у відповідні топіки.

2) Маршрутизація

Брокер приймає пакети, фільтрує їх і миттєво пересилає тим клієнтам, які підписані на ці теми.

3) Subscribe

Вебсторінка користувача є підписником. Вона постійно утримує відкрите TCP/IP з'єднання з HiveMQ. Щойно брокер отримує нові дані від датчиків, він

одразу проштовхує їх на сайт, де інтерфейс динамічно оновлюється в реальному часі.

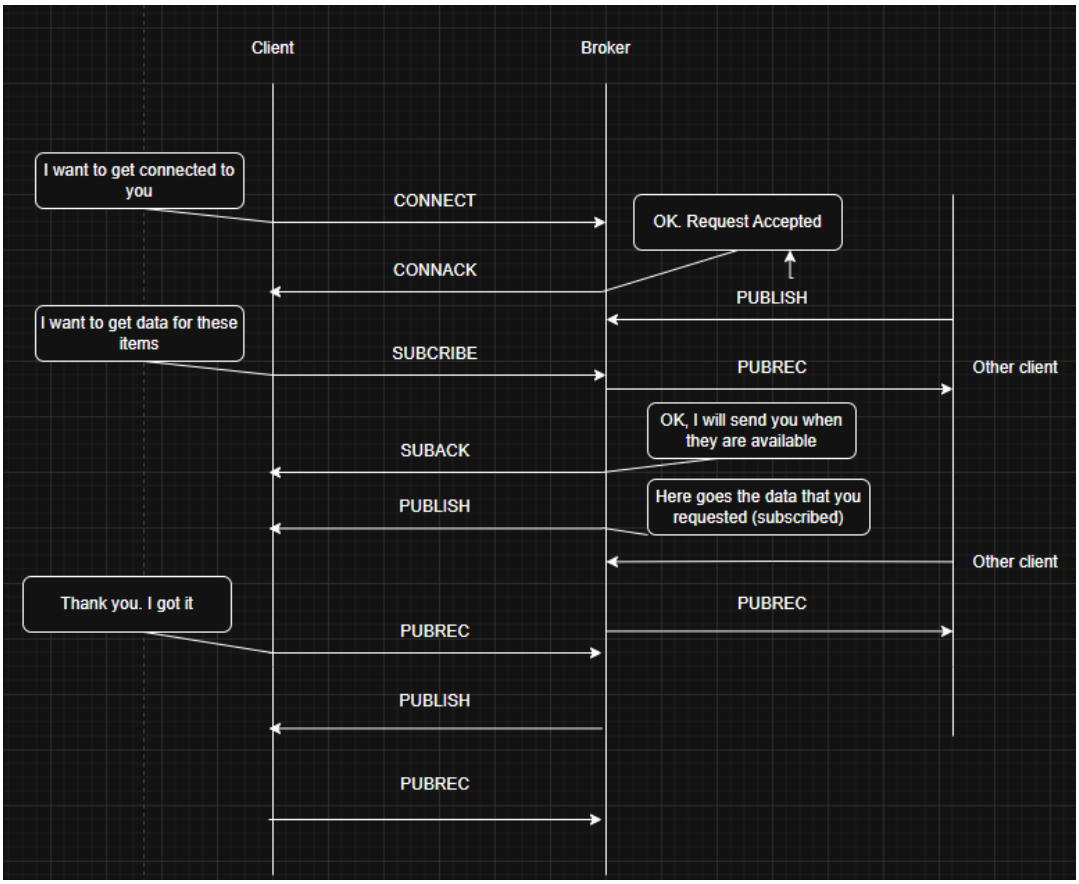

Рисунок 2.3 – Діаграма сесії зв’язку між MQTT-клієнтом та брокером

2.3 Датчик контролю температури

Температурний режим у житловому приміщенні є одним із основних факторів комфортного життя та самопочуття. Постійний моніторинг цього параметру дозволяє швидко реагувати на будь-які зміни та підтримувати комфортні та здорові умови в приміщенні.

На сьогоднішній день сучасні системи розумного будинку здійснюють контроль мікроклімату за допомогою спеціалізованих сенсорів. Одним з таких є DHT22.

Його суть роботи полягає в тому, що всередині датчика відбуваються специфічні фізичні процеси. Для вимірювання температури використовується

елемент з особливої кераміки. При підвищенні температури в кімнаті в цій кераміці зростає кількість вільних носіїв заряду, які проводять струм, що призводить до зниження електричного опору датчика, інакше кажучи коли в кімнаті стає тепліше, опір датчика падає, завдяки чому система вираховує точне значення.

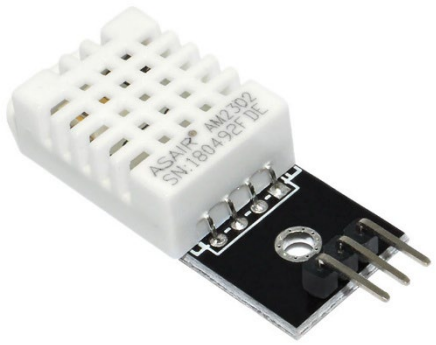

Рисунок 2.4 – Датчик температури та вологості DHT22

Хоча датчик з заводу є відкаліброваним, під час передачі даних через брокер HiveMQ на сайт все одно можуть виникати похибки. Навіть за ідеальних умов у звичайній кімнаті датчик стійку систематичну помилку і може стабільно брехати приблизно на 0,4-0,6 °C у найпопулярнішому діапазоні від 20 до 30 °C. Також його елемент вологості сильно пов’язаний із температурою , якщо казати точніше то коли в кімнаті потеплішає на 10°C, показник вологості може сам по собі зсунутися на 2-4%.

Під час досліджень, в яких показники датчика порівняно з професійним сертифікованим термогігрометром, він продемонстрував хороші результати. Показано, що середня похибка пристрою складає близько +2,99% для вологості та -2,31% для температури. Такі показники повністю відповідають офіційній документації виробника та підтверджують стабільну і передбачувальну роботу датчика [12].

Крім того, в порівнянні з іншими популярними сенсорами, обраний для проектної роботи датчик DHT22 має значно вищу точність. Наприклад, датчик

температури LM35 демонструє середню похибку близько 4,69%, а простіша модель DHT11 має значно більші відхилення, помилка його температурного режиму може змінюватися від 1% до 7% та вологості від 11% до 35%.

2.4 Датчик контролю задимленості та пожежної безпеки

Епоха електронних сенсорів почалася у 1960-х роках в Японії. Інженер Наойосі Тагучі розробив перший у світі напівпровідниковий датчик газу на основі діоксиду олова після масштабної трагедії з вибухом газу у житловому масиві [13]. Його технологія лягає в основу створення відомої лінійки датчиків серії MQ, до якої належить пристрій MQ-2 використаний у розробленій системі.

Його головне завдання полягає в тому, щоб вловлювати сліди задимлення, а також виявляти присутність небезпечних чи горючих газів таких як метан, пропан, бутан і водень. В основі роботи компонента лежить хімічна реакція – у чистому приміщенні його внутрішній напилений шар із діоксиду олова SnO_2 має високий електричний опір , що заважає проходженню струму.

Проте, коли з’являється задимлення або витік газу, ці речовини взаємодіють із розігрітою поверхнею сенсора, викликаючи різке зниження його опору. Головна плата ESP32 миттєво фіксує це падіння напруги. Аналоговий вихід дає змогу в реальному часі відслідковувати точну концентрацію забруднень у повітрі та гнучко налаштовувати чутливість системи у коді, передаючи актуальні загрози на вебсайт через HiveMQ.

Датчик диму MQ-2 має чотири виводи: VCC, GIND, AOUT і DOUT, які використовуються для зчитування інформації з сенсора:

VCC – живлення датчика, підключається до 5V;

GND – земля, підключається до GND Arduino;

DOUT – цифровий вихід: низький сигнал якщо газ не виявлено та високий при його наявності.

AOUT – аналоговий вихід, видає сигнал від 0 до 5 V залежно від концентрації газу.

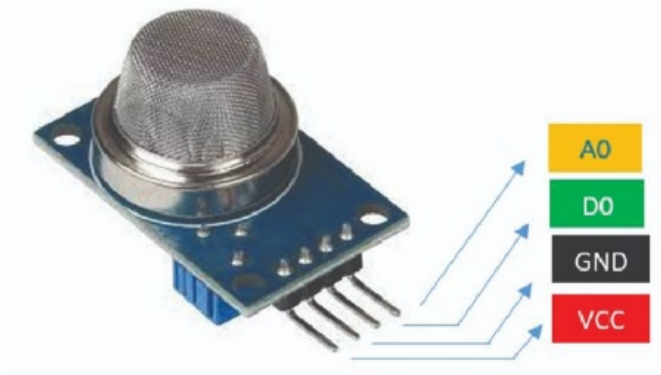

Рисунок 2.5 – Зовнішній вигляд датчика диму MQ-2

Під час інтеграції цього датчику необхідно враховувати його високе енергоспоживання. Через постійну роботу внутрішньої нитка розжарювання, яка необхідна для хімічної реакції, модуль споживає близько 150-180 мА струму і стає помітно теплим, тому живлення здійснюється виключно від 5V.

Таблиця 2.2 – Технічні параметри MQ-2

Параметр	**Опис**
Напруга живлення	5 V
Струм споживання	150 – 180 мА
Потужність нагрівального елемента	< 800 мВт
Діапазон виявлення диму та бутану	300 – 10000 ppm
Діапазон виявлення пропану/зрідженого газу	200 – 5000 ppm
Робочі температури	від -20 °C до +50 °C
Тип вихідного сигналу	Аналоговий та цифровий

Конфігурація сенсору MQ-2 показані на рис. 2.6, виробник випускає його у двох варіантах, класичному з металевою захисною сіткою для кращої конвекції повітря та компактному закритому типі. Має круглу основу діаметром приблизно 16,8 мм і оснащений шістьма симетрично розташованими металевими

виводами. Виводи поділені на дві групи – контакти фірмового нагрівального елемента та дві пари виходів з діоксиду олова, які дублюють один одного для підвищення надійності пайки.

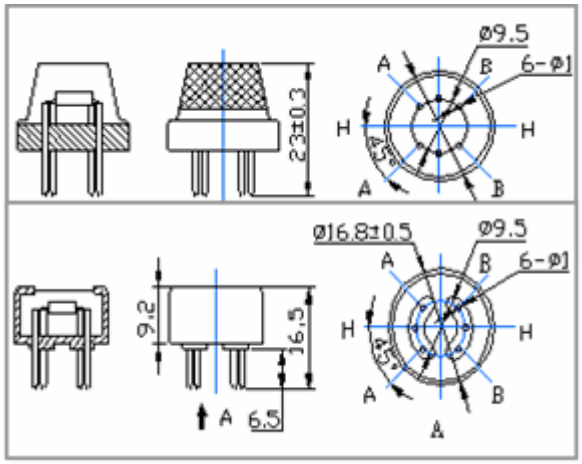

Рисунок 2.6 – Внутрішня конфігурація сенсора диму MQ-2

2.5 Датчик виявлення руху та охорони простору

Пасивні інфрачервоні датчики руху є базовим елементом сучасних систем безпеки та автоматизації завдяки їхньому надзвичайно низькому енергоспоживанню.

Традиційно в інженерії PIR-сенсори застосовуються для генерації простої наявності або відсутності руху в зоні видимості. Проте такий метод суттєво обмежує функціонал пристрою. Сучасні дослідження показують, що безпосереднє зчитування і обробка аналогового сигналу з піроелектричних елементів відкриває доступ до великого обсягу даних, таких як швидкість руху об’єкта, напрямок його переміщення, розміри та точна відстань до нього.

Принцип роботи датчика досить простий та логічний. Він фіксує інфрачервоне тепло, яке природно випромінює тіло людини або тварини. Усередині пристрою розташований чутливий кристал, поділений на кілька частин. Коли людина проходить повз датчик, їх тепло перш за все впливає на одну половину кристала, що викликає різке підвищення напруги на виході.

Потім, проходячи далі, вона зачіпає другу половину і напруга різко падає. Ці коливання напруги у вигляді чіткої синусоїдальної хвилі дозволяють платі визначити факт проходження об’єкта.

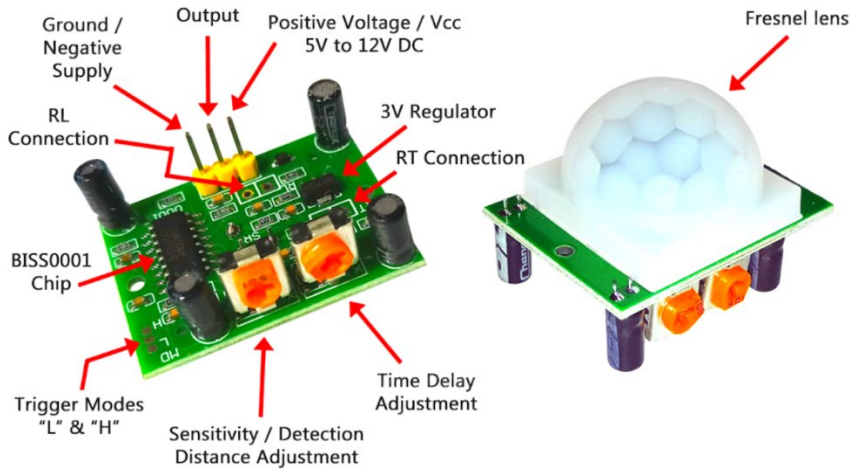

Рисунок 2.7 – Зовнішній та внутрішній вигляд датчику руху PIR

Модуль HC-SR501 (рис. 2.7) ділиться на дві частини: оптичну систему, яка збирає світло й тепло, та друковану плату з деталями для обробки сигналів.

Зовні датчик одразу впізнається за білим пластиковим куполом у формі півсфери – це лінза Френеля. Оскільки сам чутливий кристал дуже маленький і не може впіймати багато тепла, ця лінза працює як збирач. Під куполом розташована друкована плата на якій розпаяно повний комплекс елементів керування та аналізу:

- Керуюча мікросхема BISS0001 – аналого-цифровий компаратор, який є мозком датчика. Приймає слабкий первинний сигнал з кристала, підсилює його, відсікає випадковий шум і приймає рішення про видачу логічного сигналу тривоги;

- Регулятори налаштування параметрів – два вбудовані резистори з помаранчевими елементами, один відповідає за налаштування чутливості та дистанції виявлення, а інший за встановленням часу утримання сигналу тривоги;

- Конфігурація режимів тригера, спеціальна група контактів із джампером, яка дозволяє перемикати логіку роботи датчика між одиночним “L” та повторним “H” режимами;

- Інтерфейсний роз'єм зв'язку: три металеві виводи для швидкого підключення до мікроконтролерних систем, що включають контакт заземлення “Ground/Negative Supply”, цифровий вхід “Output” та лінію живлення “Positive Voltage”;

- Допоміжні лінії розширення – технологічні майданчики для встановлення фоторезистора “RT Connection” та термістора “RL Connection”, що дозволяють заблокувати роботу датчика вдень або компенсувати температурні похибки повітря;

- Внутрішній стабілізатор забезпечує роботу мікросхеми та чутливого елемента фіксованою напругою, захищаючи тракт від коливань на лінії живлення, помічається як “3V Regulator”.

Вибір датчика HC-SR501 для інтеграції в системи охорони та моніторингу простору є логічним та обґрунтованим через низку його експлуатаційних переваг. Оскільки сенсор працює в пасивному режимі і не генерує власних хвиль, його споживання струму в режимі очікування не перевищує 65 мікроампер, що робить його одним з енергоефективніших пристроїв.

Завдяки вбудованій мікросхемі BISS0001 і поворотним регуляторам на платі інженер може вручну налаштувати чутливість від 3 до 7 метрів та час утримання сигналу від 5 до 200 секунд. Це розвантажує мікроконтролер, адже йому не потрібно робити жодних складних математичних розрахунків, він просто приймає вже готовий цифровий сигнал тривоги.

2.6 Висновки розділу

У другому розділі проведено аналіз апаратного забезпечення та обґрунтовано вибір основних компонентів системи. Розглянуто архітектуру мікроконтролера ESP32, а також фізичну структуру, принцип роботи й

параметри підключених модулів: кліматичного сенсора DHT22, аналізатора повітря MQ-2 та інфрачервоного датчика руху PIR HC-SR501. Вивчення схемотехніки цих пристроїв дозволило оптимізувати зняття показників середовища.

3 МОДЕЛЮВАННЯ ТА НАЛАШТУВАННЯ СИСТЕМИ

3.1 Схема обміну даними “Edge-to-Cloud”

У кваліфікаційній роботі було спроєктовано та розроблено загальну архітектурну структуру бездротової системи моніторингу розумного будинку на базі IoT-технологій, яку наведено на рис 3.1. Представлена схема в загальному вигляді відображає наскрізну взаємодію всіх апаратних, мережевих та програмних складових проєкту. На схемі наочно показано інформаційні потоки та функціональні зв'язки між вузлами, що дає змогу чітко уявити завдання кожного рівня системи та пояснює, яким чином дані з фізичного простору кімнати потрапляють до кінцевого користувача.

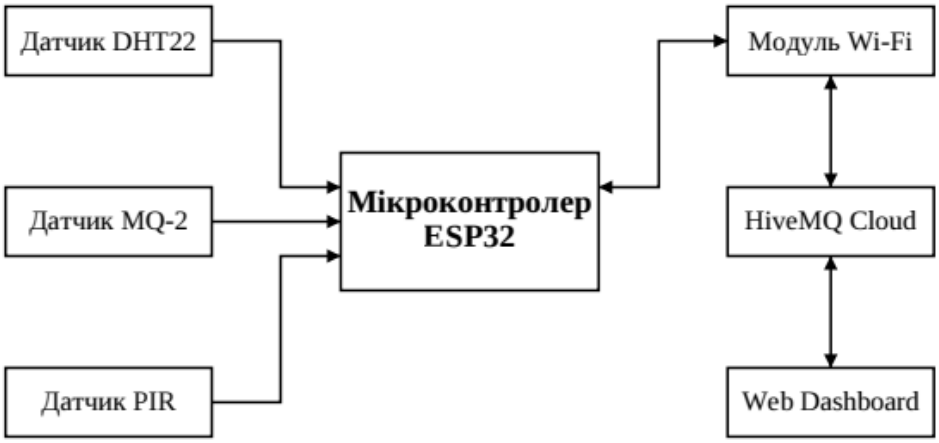

Рисунок 3.1 – Структурна схема за концепцією Edge-to-Cloud

Реалізована система є інтегрованим бездротовим IoT-рішенням, що забезпечує потоковий моніторинг і обробку даних, отриманих з датчиків у реальному часі, надаючи кінцевому користувачу гнучкий вебінтерфейс. Комплекс складається з апаратної, програмної та мережевої складових, які взаємодіють між собою через бездротове Wi-Fi з’єднання та хмарну

інфраструктуру брокера без залучення локальних реляційних баз даних та проміжних серверів.

Апаратна частина включає високопродуктивний мікроконтролер ESP32, оснащений інтегрованим радіомодулем, який забезпечує прямий зв’язок із локальною бездротовою мережею. До мікроконтролера підключені кілька спеціалізованих датчиків: цифровий сенсор DHT22 для вимірювання температури й відносної вологості, датчик MQ-2 для моніторингу рівня задимленості, а також інфрачервоний датчик руху PIR для виявлення присутності сторонніх осіб в охоронній зоні приміщення.

Дані з датчиків асинхронно зчитуються мікроконтролером, проходять первинну обробку на периферійному рівні й передаються в хмару через полегшені MQTT-публікації. Програмна частина реалізована на стороні клієнтського вебзастосунку реального часу. На відміну від традиційних систем із накопиченням інформації в таблицях MySQL, серверна логіка проєкту є повністю безсерверною та безбазовою, що мінімізує мережеві затримки.

Керування інформаційними потоками здійснюється за допомогою хмарного MQTT-брокера HiveMQ Cloud, який транслює повідомлення підписникам транзитом.

Головне вікно графічного вебінтерфейсу надає користувачеві миттєвий доступ до телеметрії об'єкта, дозволяючи переглядати поточний стан температури й вологості, динамічний рівень диму та статус безпеки простору. Вона також дозволяє візуалізувати асинхронні повідомлення про критичні події. Вебінтерфейс динамічно оновлює дані без повного перезавантаження сторінки завдяки обробці стеків повідомлень безпосередньо у браузері користувача. Мережеві компоненти системи пов’язані через TCP/IP протокол із використанням бездротового інтерфейсу Wi-Fi та прикладного протоколу MQTT.

Мікроконтролер ESP32 з високою частотою публікує актуальні параметри у відповідні топіки брокера HiveMQ, які одразу проштовхуються на сайт, де інтерфейсна частина на базі мови JavaScript динамічно зчитує та відображає пакети телеметрії в режимі реального часу.

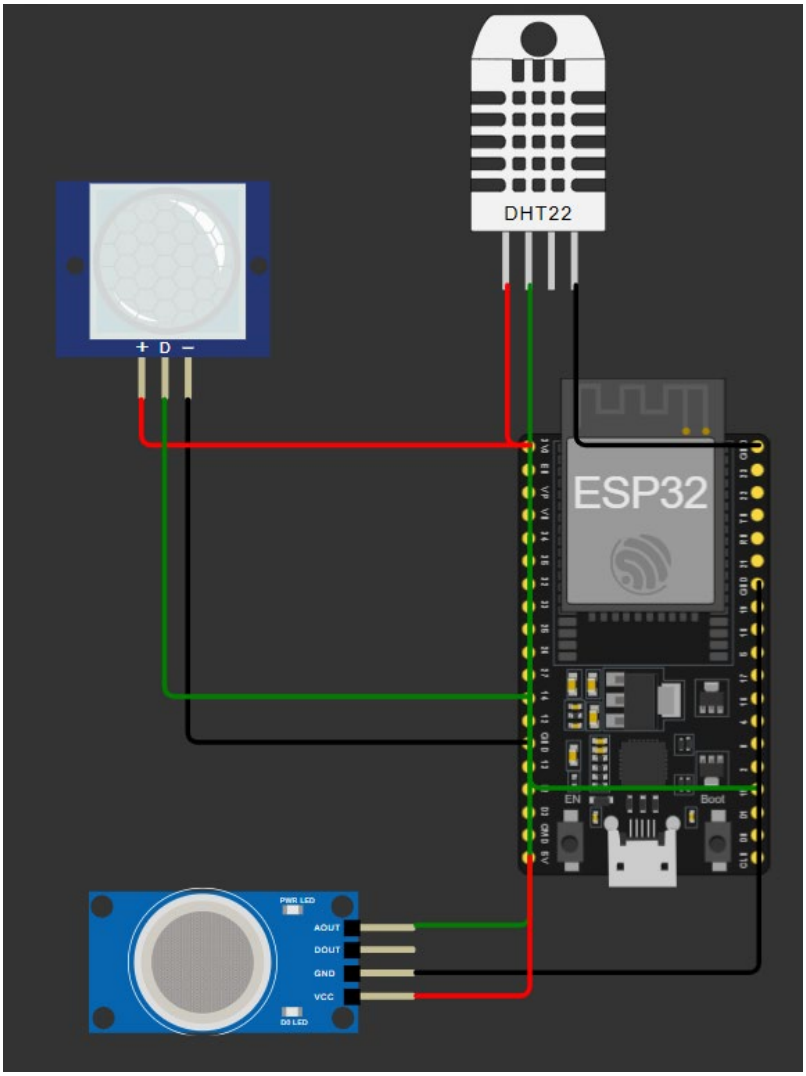

Рисунок 3.2 – Функціональна схема у середовищі Wokwi

Для контролю рівня задимленості використовується датчик MQ-2. Його аналоговий вихід AOUT підключений до порту GPIO34 на платі ESP32. Провід живлення датчика VCC підключено до контакту 5V, а провід заземлення до контакту GND на мікроконтролері.

Для виявлення руху в кімнаті використовується датчик руху PIR. Його живлення підключено до контакту 3,3V, заземлення до контакту GND, а вихідний сигнальний провід OUT підключений до цифрового входу GPIO14 на платі ESP32.

Датчик температури та вологості DHT22 використовується для моніторингу клімату. Його живлення підключено до контакту безпечної напруги

3.3V, заземлення зведено на спільну шину GND, а сигнальний вивід SDA підключений до цифрового входу GPIO15 для передачі інформації за однодротовим інтерфейсом.

3.2 Організація потоків телеметрії

Система працює без традиційних баз даних і окремого сервера для зберігання історії. Тому було налаштовано передачу даних безпосередньо на рівні мікроконтролера ESP32. Щоб система працювала стабільно і без збоїв, було розділено адресний простір у хмарному MQTT-брокері HiveMQ Cloud. Крім того, встановлено різні пріоритети для відправки повідомлень залежно від того, який датчик їх надсилає.

Кожен із трьох датчиків отримав власний топік (DHT22 отримав 2 топіки) у просторі MQTT-брокера HiveMQ Cloud, побудований за ієрархічним принципом “розділ/тип даних”, що спрощує підписку клієнтської частини на конкретні показники без потреби розбирати змішаний потік повідомлень. Для кліматичних параметрів обрано інтервал публікації, що не перевищує кожні п’ять секунд, оскільки температура та вологість у житловому приміщенні змінюються поступово, і частіші вимірювання лише марно навантажували б канал зв’язку.

3.3 Вибір середовища моделювання та розробки системи

Використання реального обладнання на ранніх етапах розробки є економічно та організаційно невигідним: помилка в схемі підключення чи в конфігурації GPIO може призвести до пошкодження мікроконтролера або периферійних модулів, а процес закупівлі й заміни компонентів суттєво уповільнює розробку прошивки. З огляду на це було обрано онлайн-симулятор Wokwi, який дозволяє відтворити віртуальну електричну схему на базі мікроконтролера ESP32 та підключених до нього датчиків без необхідності мати фізичну форму.

Wokwi надає готову бібліотеку віртуальних компонентів, серед яких присутні ті модулі, що використовуються в розробленій системі: датчик DHT22, газовий сенсор MQ-2 та датчик руху HC-SR501, а також підтримує повноцінне виконання прошивки. Завдяки цьому можна перевірити реальну логіку роботи програми: обробку переривань, формування JSON-структур, публікацію повідомлень у топіки MQTT-брокера та реакцію системи на зміну показань датчиків у часі.

Симулятор інтегрується безпосередньо в середовище розробки Visual Studio Code через відповідне розширення, що дозволяє редагувати код, компілювати прошивку та одразу запускати симуляцію в межах одного робочого простору, без перемикання між окремими застосунками.

Для написання та збирання самої прошивки мікроконтролера обрано зв’язку Visual Studio Code як текстовий редактор та PlatformIO як платформу для керування залежностями й процесом компіляції.

3.4 Підготовка програмного середовища та бібліотек для ESP32

Створення проєкту розпочалося з формування нового середовища PlatformIO, у якому обрано плату ESP32, а як фреймворк розробки: Arduino Framework. Такий вибір фреймворку обумовлений насамперед зручністю: Arduino Framework надає високорівневий та добре задокументований набір функцій для роботи.

Після ініціалізації проєкту PlatformIO автоматично формує типову структуру каталогів, яка включає теку “src” з основним файлом прошивки “main.cpp”, теку “include” для заголовкових файлів проєкту, теку “lib” для локальних бібліотек та конфігураційний файл “platformio.ini”, що визначає параметри збирання: тип плати, фреймворк, швидкість послідовного порту та перелік підключених бібліотек.

Підключення зовнішніх бібліотек здійснено через секцію “lib_deps” файлу platformio.ini, що дозволяє PlatformIO самостійно завантажувати потрібні версії пакетів під час першого збирання проєкту. Для реалізації бездротового з’єднання

використано вбудовану бібліотеку “WiFi.h”, яка входить до складу Arduino Framework для ESP32 та надає прямий доступ до апаратного радіомодуля мікроконтролера без потреби у сторонніх драйверах. Для роботи з протоколом MQTT обрано бібліотеку PubSubClient.

3.5 Висновки розділу

У третьому розділі кваліфікаційної роботи було успішно виконано повний цикл інженерного проєктування, апаратно-програмної реалізації та комп'ютерного моделювання бездротової системи моніторингу житлового простору. На основі концепції Edge-to-Cloud розроблено децентралізовану структуру інформаційних потоків, яка пов'язує периферійний контролер ESP32, хмарний брокер HiveMQ Cloud та адаптивний вебінтерфейс реального часу. Головною перевагою створеної архітектури є повна відмова від використання традиційних серверних баз даних, що дозволило усунути затримки на операції запису й зчитування, знизити апаратні вимоги до вузлів та забезпечити транзитну передачу телеметрії безпосередньо у браузер користувача.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ВЕБІНТЕРФЕЙС СИСТЕМИ

4.1 Розробка керуючого коду ESP32 на C++

Програмну частину системи реалізовано мовою програмування C++ із використанням середовища PlatformIO. Основний програмний код розміщено у файлі “main.cpp” (Додаток Б).

На початковому етапі програми підключаються бібліотеки, необхідні для роботи основних функціональних модулів системи:

Лістинг 4.1 – Підключення бібліотек

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <DHT.h>
```

Бібліотека “WiFi.h” використовується для підключення ESP32 до бездротової мережі, “WiFiClientSecure.h” забезпечує створення захищеного мережевого з’єднання з хмарним сервером HiveMQ Cloud. Для реалізації обміну повідомленнями за протоколом MQTT використано бібліотеку “PubSubClient.h”, “DHT.h” відповідає за коректне зчитування температури та вологості з датчика DHT22.

Датчики PIR та MQ-2 не потребують окремих бібліотек, оскільки їхні сигнали зчитуються стандартними засобами Arduino Framework.

Після підключення бібліотек визначаються контакти мікроконтролера, до яких підключені сенсори, а також основні параметри MQTT-з’єднання. Це дозволяє одразу описати апаратну та мережеву конфігурацію системи:

Лістинг 4.2 – Конфігурація датчиків та MQTT-топіків

```
#define DHTPIN 15
#define DHTTYPE DHT22
```

Продовження лістингу 4.2

```
#define PIRPIN 14
#define MQ2PIN 34

// ===== Wi-Fi для Wokwi =====
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// ===== HiveMQ Cloud MQTT =====

const char* mqtt_server =
"f94e3665ef9547478351393dce03faf3.s1.eu.hivemq.cloud";
const int mqtt_port = 8883; // для ESP32 MQTT
const char* mqtt_user = "suhotenko_user";
const char* mqtt_pass = "SuhotenkoPass123";

// ===== MQTT topics =====

const char* topic_temp = "suhotenko/home/temperature";
const char* topic_hum  = "suhotenko/home/humidity";
const char* topic_gas  = "suhotenko/home/gas";
const char* topic_pir  = "suhotenko/home/motion";
```

У наведеному фрагменті показано, що датчик DHT22 підключено до GPIO15, PIR-сенсор до GPIO14, а аналоговий вихід датчика MQ-2 до GPIO34. Таке розділення контактів дозволяє одночасно отримувати цифрові та аналогові сигнали від різних типів сенсорів.

Для передавання телеметрії використано окремі MQTT-топіки. Температура, вологість, рівень газу та стан датчика руху передаються окремими каналами. Завдяки цьому вебзастосунок може незалежно підписуватися на потрібні повідомлення та оновлювати відповідні елементи інтерфейсу без додаткової обробки складних пакетів даних.

Початкова ініціалізація системи виконується у стандартній функції “setup()”, яка запускається один раз після старту мікроконтролера:

Лістинг 4.3 – Ініціалізація системи

```
void setup() {
  Serial.begin(115200);

  pinMode(PIRPIN, INPUT);
  pinMode(MQ2PIN, INPUT);

  dht.begin();
  setup_wifi();
  espClient.setInsecure();
  client.setServer(mqtt_server, mqtt_port);
}
```

У функції “setup()” спочатку запускається послідовний порт зі швидкістю 115200 бод. Він використовується для виведення повідомлень під час тестування системи. Далі контакти PIR та MQ-2 налаштовуються як входи, оскільки ESP32 отримує від них інформаційні сигнали. Після цього виконується ініціалізація датчика DHT22 за допомогою команди “dht.begin()”.

Функція “setup_wifi()” відповідає за підключення плати ESP32 до бездротової мережі. Команда “espClient.setInsecure()” використовується для спрощення TLS-з’єднання з HiveMQ Cloud в межах навчального та симуляційного проєкту. Після цього за допомогою “client.setServer()” задаються адреса та порт MQTT-брокера.

Після підготовки мережевого середовища програма повинна встановити з’єднання зброкером. Для цього використовується механізм автентифікації за іменем користувача та паролем:

Лістинг 4.4 – Підключення до MQTT-брокера

```
  if (client.connect(clientId.c_str(), mqtt_user, mqtt_pass))
{
      Serial.println("успішно");
    } else {
      Serial.print("помилка, rc=");
      Serial.print(client.state());
      Serial.println(" повтор через 5 секунд");
      delay(5000);
    }
```

У цьому фрагменті програми виконується підключення ESP32 до хмарного MQTT-брокера HiveMQ Cloud із використанням попередньо сформованого ідентифікатора клієнта “clientId”, а також облікових даних користувача (“mqtt_user” та “mqtt_pass”). Метод “client.connect()” повертає логічне значення, що дозволяє визначити, чи було успішно встановлено з'єднання. У разі виникнення помилки програма виводить код помилки за допомогою функції “client.state()”.

Після встановлення з’єднання мікроконтролер переходить до основного етапу роботи - зчитування показників із підключених датчиків:

Лістинг 4.5 – Зчитування даних з сенсора

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
int gasValue = analogRead(MQ2PIN);
int motion = digitalRead(PIRPIN);
```

У цьому фрагменті з датчика DHT22 отримуються два параметри: температура та відносна вологість повітря. Значення температури зберігається у змінній “temperature”, а значення вологості у змінній “humidity”.

Датчик MQ-2 підключено до аналогового входу мікроконтролера, тому його показник зчитується функцією “analogRead()”. Отримане значення відображає рівень сигналу, який змінюється залежно від концентрації диму або газу в повітрі. PIR-сенсор зчитується функцією “digitalRead()" і повертає логічний стан: наявність або відсутність руху в зоні контролю.

Перед передаванням до брокера отримані значення перетворюються у текстовий формат даних та публікуються у відповідні топіки:

Лістинг 4.6 – Передавання телеметричних даних через MQTT

```
client.publish(topic_temp, tempStr);
client.publish(topic_hum, humStr);
client.publish(topic_gas, gasStr);
client.publish(topic_pir, pirStr);
```

Команда “client.publish()” надсилає повідомлення до MQTT-брокера.

Кожен параметр передається в окремий топік: температура в suhotenko/home/temperature, вологість в suhotenko/home/humidity, рівень газу в suhotenko/home/gas, а стан руху в suhotenko/home/motion.

4.2 Програмне проєктування вебзастосунку

Наступним етапом роботи стало створення вебзастосунку, призначеного для відображення телеметричних даних у режимі реального часу. Основним завданням інтерфейсу є забезпечення зручного моніторингу параметрів житлового середовища без необхідності використання спеціалізованого програмного забезпечення або додаткових клієнтських застосунків.

Вебзастосунок розроблено із використанням сучасних технологій HTML5, CSS3 та JavaScrip.

Архітектура вебзастосунку побудована таким чином, що сторінка безпосередньо підключається до MQTT-брокера HiveMQ Cloud за допомогою WebSocket-з'єднання. Після встановлення з'єднання застосунок автоматично підписується на відповідні топіки та приймає повідомлення від мікроконтролера ESP32. Отримані значення температури, вологості, концентрації газу та стану датчика руху миттєво відображаються у графічному інтерфейсі без необхідності перезавантаження сторінки.

Структурно вебзастосунок складається з трьох компонентів: HTML-документа, CSS-файлу та JavaScript-модуля, який реалізує взаємодію з MQTT-брокером і динамічне оновлення даних.

4.2.1 Реалізація інтерфейсу в HTML

У верхній частині HTML-документа розташовано службові елементи, які визначають тип документа, кодування символів, назву вебсторінки та підключення таблиці стилів:

Лістинг 4.7 – Початкова структура HTML-документа

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Smart Home IoT</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
```

Директива <!DOCTYPE html> повідомляє браузеру про використання стандарту HTML5. Атрибут “lang="uk"” визначає українську мову документа, а тег “meta” встановлює кодування UTF-8 для коректного відображення національних символів. За допомогою елемента “link” підключається зовнішній файл стилів, який відповідає за оформлення вебінтерфейсу.

Після службової частини формується основний контейнер сторінки, у якому розміщуються всі інформаційні блоки системи моніторингу:

Лістинг 4.8 – Формування інформаційних карток вебзастосунку

```
<body>
  <div class="container">
    <h1>Система моніторингу розумного будинку</h1>
    <div class="cards">
      <div class="card">
        <h2>Температура</h2>
        <p id="temperature">-- °C</p>
      </div>
      <div class="card">
        <h2>Вологість</h2>
        <p id="humidity">-- %</p>
      </div>
      <div class="card">
        <h2>Рівень газу / диму</h2>
        <p id="gas">--</p>
      </div>
      <div class="card">
        <h2>Рух</h2>
```

Продовження лістингу 4.8

```
        <p id="motion">--</p>
      </div>
    </div>
    <p id="status">Підключення до MQTT...</p>
  </div>.
```

Центральним елементом інтерфейсу є контейнер “container”, усередині якого розміщено заголовок застосунку та чотири інформаційні картки. Кожна картка призначена для відображення окремого параметра системи: температури повітря, вологості, рівня газу або диму та стану датчика руху.

Для кожного інформаційного поля використано власний ідентифікатор (temperature, humidity, gas, motion). Надалі саме ці елементи змінюються JavaScript-кодом після отримання MQTT-повідомлень від мікроконтролера ESP32.

У нижній частині сторінки розташовано текстовий індикатор стану підключення до MQTT-брокера, який інформує про процес встановлення або втрати мережевого з'єднання.

Після формування структури документа виконано підключення необхідних JavaScript-файлів:

Лістинг 4.9 – Підключення модулів JavaScript

```
<script src="https://unpkg.com/mqtt/dist/mqtt.min.js"></script>
<script src="script.js"></script>
    </body>
    </html>
```

Спочатку підключається бібліотека “MQTT.js”, яка реалізує клієнт MQTT поверх WebSocket-з'єднання та забезпечує взаємодію вебзастосунку із брокером. Після цього підключається файл “script.js”, у якому реалізовано логіку підключення до MQTT-сервера, підписку на необхідні топіки та автоматичне оновлення показників сенсорів у режимі реального часу.

4.2.2 Стилізація графічного вікна в CSS

Наступним етапом стала розробка візуального оформлення вебсторінки. Використання окремого файлу style.css забезпечує відокремлення логіки представлення від структури HTML-документа, що спрощує подальшу модифікацію та супровід програмного забезпечення.

Було визначено базові стилі сторінки, які задають загальний вигляд вебзастосунку:

Лістинг 4.10 – Оформлення вебсторінки

```
body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: #eef2f5;
}
.container {
  max-width: 1000px;
  margin: 40px auto;
  text-align: center;
}
h1 {
  margin-bottom: 30px;
}
```

Основною частиною інтерфейсу є інформаційні картки, у яких відображаються показники сенсорів системи:

Лістинг 4.11 – Стилізація інформаційних карток

```
.cards {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 20px;
}
```

Продовження лістингу 4.11

```
.card {
  background: white;
  padding: 25px;
  border-radius: 15px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.15);
}
```

Для розташування блоків використано сучасний механізм CSS Grid Layout, який дозволяє автоматично формувати чотири рівномірні колонки. Завдяки властивості “gap” між картками створюється однаковий інтервал, що робить інтерфейс більш впорядкованим.

Завершальним етапом оформлення стало налаштування текстових елементів і статусного повідомлення про стан MQTT-з'єднання:

Лістинг 4.12 – Оформлення текстових елементів

```
.card h2 {
  font-size: 18px;
  margin-bottom: 15px;
}
.card p {
  font-size: 28px;
  font-weight: bold;
}
#status {
  margin-top: 30px;
  font-weight: bold;
}
```

Для заголовків інформаційних карток встановлено розмір шрифту 18 пікселів, що забезпечує їх чітке відокремлення від основних показників. Значення температури, вологості, концентрації газу та стану датчика руху відображаються збільшеним напівжирним шрифтом.

Окрему увагу приділено текстовому індикатору “status”, який використовується для інформування користувача про стан підключення до MQTT-брокера.

4.2.3 Динамічне приймання MQTT-повідомлень на JavaScript

Основним призначенням файлу “script.js” є встановлення з'єднання з MQTT-брокером HiveMQ Cloud, підписка на телеметричні топіки та динамічне оновлення значень у графічному інтерфейсі користувача:

Лістинг 4.13 – Налаштування підключення до брокера

```
const client =
mqtt.connect("wss://f94e3665ef9547478351393dce03faf3.s1.eu.hiv
emq.cloud:8884/mqtt", {
  username: "suhotenko_user",
  password: "SuhotenkoPass123",
```

Продовження лістингу 4.13

```
clientId: "web_client_" + Math.random().toString(16).substr(2,
8),
  protocolVersion: 4,
  clean: true,
  reconnectPeriod: 3000,
  connectTimeout: 30 * 1000
});
```

У цьому фрагменті використовується метод “mqtt.connect()”, який створює з'єднання між браузером та брокером. Адреса брокера починається з префікса wss://, що означає використання захищеного WebSocket-з'єднання. Порт 8884 застосовується для підключення вебклієнта до HiveMQ Cloud через WebSocket.

Параметри “username” та “password” використовуються для автентифікації клієнта на брокері. Параметр “clientId” формується автоматично з випадковим фрагментом, що дозволяє уникнути конфлікту між одночасно відкритими вебсторінками.

Після створення MQTT-клієнта у програмі визначається список топіків, на які має підписатися вебзастосунок:

Лістинг 4.14 – Формування списку топіків

```
const topics = [
  "suhotenko/home/temperature",
  "suhotenko/home/humidity",
  "suhotenko/home/gas",
  "suhotenko/home/motion"
];
```

Основна логіка оновлення інтерфейсу реалізована в обробнику події “message”. Ця подія спрацьовує кожного разу, коли брокер передає браузеру нове MQTT-повідомлення:

Лістинг 4.15 – Приймання повідомлень та оновлення HTML-елементів

```
client.on("message", (topic, message) => {
  const value = message.toString();
  console.log(topic, value);
  if (topic === "suhotenko/home/temperature") {
    document.getElementById("temperature").innerText = value + "
°C";
  }
  if (topic === "suhotenko/home/humidity") {
    document.getElementById("humidity").innerText = value + " %";
  }
  if (topic === "suhotenko/home/gas") {
    document.getElementById("gas").innerText = value;
  }
  if (topic === "suhotenko/home/motion") {
    document.getElementById("motion").innerText =
      value === "1" ? "Рух виявлено" : "Руху немає";
  }
});
```

Кожне отримане MQTT-повідомлення перетворюється у текстовий формат за допомогою “toString()”. Далі програма перевіряє, з якого саме топіка надійшло повідомлення, і відповідно оновлює потрібний елемент сторінки.

Для підвищення надійності роботи застосунку також реалізовано обробку помилок і втрати з'єднання з MQTT-брокером:

Лістинг 4.16 – Обробка помилок та відключення з’єднання

```
client.on("error", (error) => {
  console.log("MQTT error:", error);
  document.getElementById("status").innerText = "Помилка
MQTT";
});
client.on("close", () => {
  console.log("MQTT з'єднання закрито");
          document.getElementById("status").innerText =
"MQTT відключено";
});
```

Подія “error” спрацьовує у випадку виникнення помилки під час підключення або обміну повідомленнями. У такій ситуації повідомлення про помилку виводиться в консоль браузера, а на сторінці змінюється статус підключення. Подія “close” викликається тоді, коли MQTT-з'єднання було закрито.

4.3 Тестування та перевірка розробленої системи

Після завершення розробки програмного забезпечення було виконано перевірку працездатності інформаційно-керуючої системи. Тестування здійснювалося у середовищі Wokwi із використанням MQTT-брокера HiveMQ Cloud та розробленого вебзастосунку.

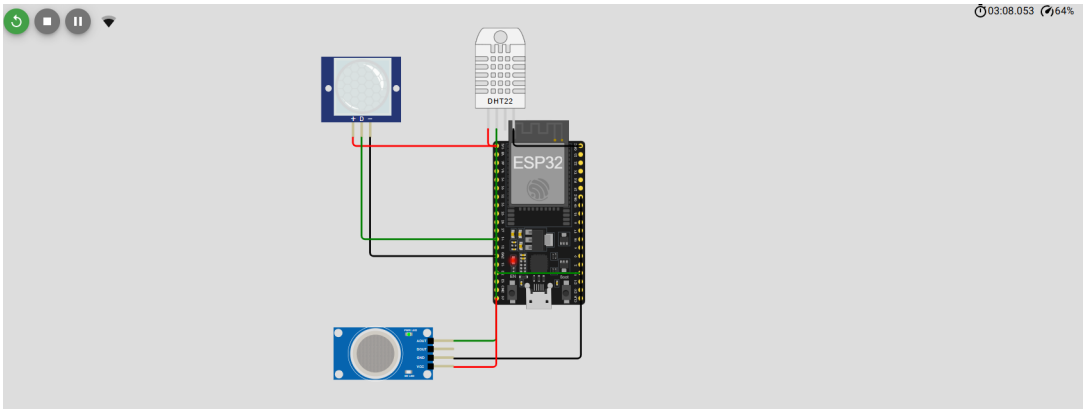

Рисунок 4.1 – Робота моделі ESP32 у середовищі Wokwi

Після запуску моделі ESP32 успішно підключалася до мережі Wi-Fi та ініціалізувала підключені датчики DHT22, MQ-2 і PIR. Зчитування показників виконувалося автоматично через кожні три секунди.

Для перевірки роботи MQTT-протоколу використовувався сервіс HiveMQ Cloud. У вкладці Messages підтверджується надходження повідомлень у всі створені топіки, що свідчить про коректну передачу телеметричних даних від ESP32 (рис. 4.2).

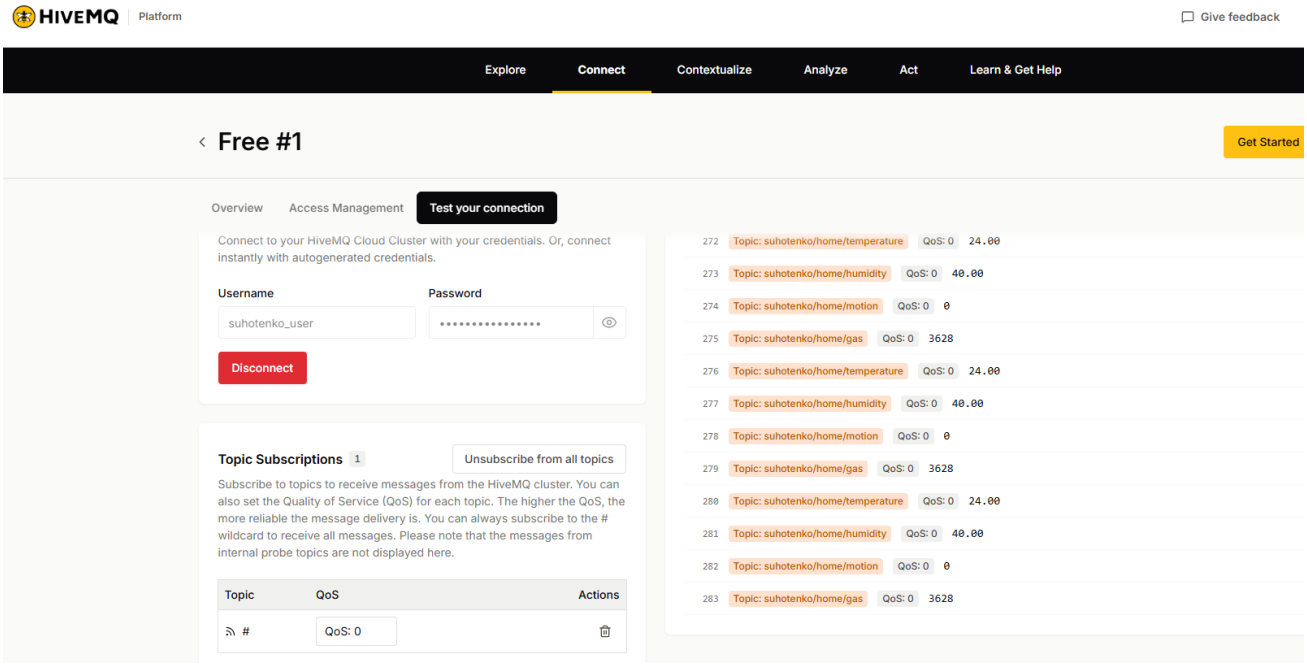

Рисунок 4.2 – Передавання повідомлень у HiveMQ Cloud

Для перевірки роботи клієнтської частини системи головну вебсторінку було відкрито через локальний сервер Live Server за адресою 127.0.0.1:5500/index.html. Після встановлення з'єднання з брокером HiveMQ Cloud вебзастосунок автоматично підписувався на відповідні MQTT-топіки та відображав актуальні показники температури, вологості, рівня газу та стану датчика руху.

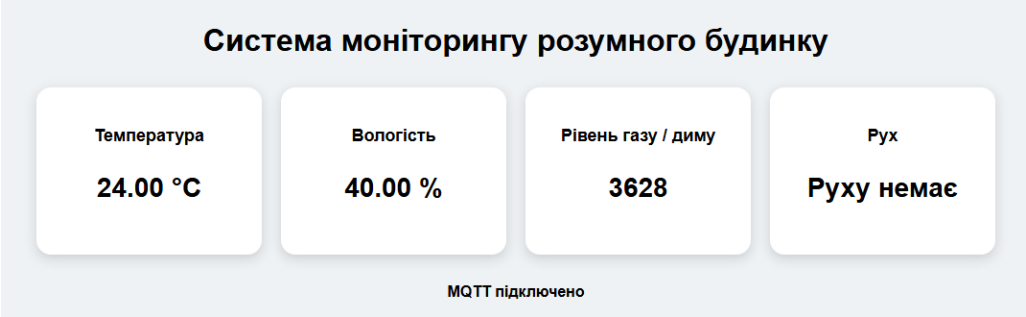

Рисунок 4.3– Головне вікно вебзастосунку

Отримані від мікроконтролера ESP32 дані оновлюються на сторінці в режимі реального часу без необхідності її перезавантаження, що підтверджує коректну взаємодію між апаратною та програмною частинами розробленої інформаційно-керуючої системи.

Під час проведення тестування було встановлено, що всі значення, отримані від підключених датчиків, своєчасно передаються через MQTT-брокер та коректно відображаються у вебінтерфейсі. Затримка між зміною показників датчиків і їх появою на сторінці є мінімальною, що забезпечує оперативний моніторинг параметрів житлового середовища.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досліджено сучасні підходи до побудови інформаційно-керуючих систем житлового середовища на базі технологій Інтернету речей. Проведено аналіз принципів функціонування IoT-систем, їх архітектури, мережевих протоколів та сучасних платформ автоматизації житла. Особливу увагу приділено використанню протоколу MQTT для обміну телеметричними даними у режимі реального часу.

На основі проведеного аналізу було обґрунтовано вибір апаратної та програмної платформи для реалізації проєкту. Як центральний елемент системи використано мікроконтролер ESP32 із вбудованим модулем Wi-Fi, який забезпечує бездротове підключення до мережі та підтримує роботу з хмарним MQTT-брокером HiveMQ Cloud. Для моніторингу стану житлового середовища до системи інтегровано цифровий датчик температури та вологості DHT22, датчик концентрації газу MQ-2 і пасивний інфрачервоний датчик руху PIR.

У роботі розроблено програмне забезпечення мікроконтролера мовою програмування C++, яке реалізує зчитування показників із сенсорів, обробку отриманих даних та їх публікацію до MQTT-брокера через захищене мережеве з'єднання. Для візуалізації телеметричної інформації створено вебзастосунок на основі технологій HTML, CSS та JavaScript, який забезпечує підключення до HiveMQ Cloud, автоматичну підписку на необхідні MQTT-топіки та відображення отриманих даних у режимі реального часу.

Розроблена інформаційно-керуюча система забезпечує дистанційний моніторинг основних параметрів житлового середовища та може використовуватися як основа для подальшого розвитку комплексних систем розумного будинку. Перспективними напрямами вдосконалення є інтеграція виконавчих пристроїв для автоматичного керування обладнанням, реалізація системи сповіщень користувача про аварійні ситуації, розширення переліку підключених датчиків, а також впровадження сучасних механізмів захисту інформації та автентифікації користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bhat O. Introduction to IoT [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/profile/Omkar-Bhat/publication/330114646_Introduction_to_IOT/links/5c2e31cf299bf12be3ab21eb/Introduction-to-IOT.pdf
2. Internet of Things: Vision, Applications and Research Challenges [Електронний ресурс]. – Режим доступу: https://storage.prod.researchhub.com/uploads/papers/2022/01/11/JCC_20150525160139 23_wWC8gIW.pdf
3. Інтернет речей та сфера його застосування [Електронний ресурс]. – Режим доступу: https://server-shop.ua/the-internet-of-things-and-the-scope-of-its-use.html
4. Alaa M., Zaidan A. A., Zaidan B. B. та ін. A Review of Smart Home Applications Based on Internet of Things // Sensors. – 2020. – Vol. 20, № 13. – Art. 3625.
5. Інтернет речей: сучасний стан, проблеми та перспективи розвитку [Електронний ресурс]. – Режим доступу: https://openarchive.nure.ua/entities/publication/65cb2231-d9c9-4ec5-a255-352cc63af165
6. Інтернет речей: принципи побудови та перспективи використання [Електронний ресурс]. – Режим доступу: https://dspace.onu.edu.ua/server/api/core/bitstreams/2a9d4f98-53ac-4154-b238-2221cd6b0453/content
7. Goodbye Legacy HomeKit, Hello Apple Home [Електронний ресурс]. – Режим доступу: https://www.applehomeauthority.com/goodbye-legacy-homekit-hello-apple-home/
8. Xiaomi IoT Platform [Електронний ресурс]. – Режим доступу: https://iot.mi.com/
9. Kurniawan B. Introduction to ESP32 // Beginning ESP32 Application Development with the ESP-IDF. – Berkeley : Apress, 2023. – Chapter 1.

10. Повний посібник з мікроконтролера ESP32 [Електронний ресурс]. – Режим доступу: https://ua.ariat-tech.com/blog/ESP32-Microcontroller-Guide.html
11. Soni D., Makwana A. A Survey on MQTT: A Protocol of Internet of Things (IoT) // International Journal of Engineering Research and Development. – 2017.
12. Analisis Akurasi Sistem Sensor DHT22 Berbasis Arduino [Електронний ресурс]. – Режим доступу: https://d1wqtxts1xzle7.cloudfront.net/96786036/290094716-libre.pdf
13. History of FIGARO Engineering Inc. [Електронний ресурс]. – Режим доступу: https://www.figaro.co.jp/en/company/history.html
14. Verma P., Sood S. K. Cloud-centric IoT based smart home framework for enhanced energy management // Internet of Things. – 2020. – Vol. 10. – Art. 100177.