НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету Інформаційних технологій	**Завідувач кафедри** Комп'ютерних систем, мереж та кібербезпеки
_____Ігор БОЛБОТ (підпис)	_____Дмитро КАСАТКІН (підпис)
«___»_____2026 р.	«___»_____2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: «Розробка бездротової мережі для моніторингу панелей у СЕС »

Спеціальність F7 «Комп'ютерна інженерія »

Освітньо-професійна програма «Комп'ютерна інженерія »

Гарант освітньої програми
канд. фіз.-мат. наук, доцент _____ (підпис) **Євгеній НІКІТЕНКО**

Керівник бакалаврської кваліфікаційної роботи
канд. техн. наук, доцент _____ (підпис) **Віктор СМОЛІЙ**

Виконав _____ (підпис) **Артем МІХАЛЕВСЬКИЙ**

КИЇВ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
канд . пед. наук, доцент_____Дмитро КАСАТКІН
«_____»_____20____
р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Міхалевському Артему Андрійовичу
(прізвище, ім'я, по батькові)

Спеціальність «Комп'ютерна інженерія »

Освітньо-професійна програма «Комп'ютерна інженерія »

Тема кваліфікаційної бакалаврської роботи
«Розробка бездротової мережі для моніторингу панелей у СЕС
»

затверджена наказом ректора НУБіП України від «10»__12_____2025 р. №_3072 «С»___
Термін подання завершеної роботи на кафедру «22»_05_2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи
Мікроконтролер — ESP32, технологія бездротової мережі — WiFi,
Масштаб СЕС — приватні, потужністю кілька десятків кВт.
Перелік питань, що підлягають дослідженню:

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ
2. ПРОЄКТУВАННЯ БЕЗДРОТОВОЇ СИСТЕМИ МОНІТОРИНГУ ПАНЕЛЕЙ У СЕС
3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ НА БАЗІ ESP32
4. МОДЕЛЮВАННЯ, НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ СИСТЕМИ

Перелік графічного матеріалу (за необхідності).

Дата видачі завдання "__10__"_____12_____2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ **Віктор СМОЛІЙ**
канд. техн. наук, доцент (підпис)

Завдання взяв до виконання _____ **Артем МІХАЛЕВСЬКИЙ**
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз предметної області	04.01.2026 р.	Виконано
2	Проектування системи	15.02.2026 р.	Виконано
3	Реалізація системи	10.03.2026 р.	Виконано
4	Тестування системи	01.04.2026 р.	Виконано
5	Оформлення пояснювальної записки	01.05.2026 р.	Виконано
6	Оформлення графічного матеріалу	01.05.2026 р.	Виконано

Студент _____ **А.А. Міхалевський**

(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **В.В СМОЛІЙ**

(підпис) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 83 сторінок, 13 рисунків, 2 таблиці, 11 лістингів, 5 додатків, 12 джерел.

БЕЗДРОТОВА СИСТЕМА МОНІТОРИНГУ, СОНЯЧНА ЕЛЕКТРОСТАНЦІЯ, ESP32, MQTT, HTTP, JSON, WOKWI, КОНТРОЛЕР-ШЛЮЗ, IoT

Об’єкт дослідження - процес бездротового моніторингу параметрів сонячних панелей у складі сонячної електростанції.

Предмет дослідження - апаратні та програмні засоби побудови багатовузлової системи збору, обробки та передавання даних про стан панелей СЕС.

Мета роботи - розробка бездротової системи моніторингу панелей у СЕС на базі мікроконтролерів ESP32 з передаванням даних до контролера-шлюзу та подальшим надсиланням інформації на сервер.

У роботі розглянуто особливості моніторингу сонячних електростанцій, визначено основні параметри контролю панелей, обґрунтовано вибір апаратних і програмних засобів. Розроблено структуру системи, яка складається з вузлів моніторингу панелей, контролера-шлюзу та серверної частини.

У програмній частині реалізовано зчитування температури, освітленості, напруги та струму, обчислення потужності панелі, формування JSON-повідомлень і передавання телеметрії через MQTT. Контролер-шлюз приймає дані від вузлів, виконує їх обробку, зберігає поточний стан панелей і надсилає інформацію на сервер за допомогою HTTP POST.

Перевірку роботи системи виконано в середовищі Wokwi. У результаті моделювання підтверджено можливість передавання даних від вузлів панелей до контролера-шлюзу та від шлюзу до серверної частини. Практичний результат роботи полягає у створенні працездатної моделі бездротової системи моніторингу, яку можна використовувати як основу для подальшого розширення та адаптації до реального обладнання.

Зміст

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

ADC - Analog-to-Digital Converter, аналого-цифровий перетворювач

API - Application Programming Interface, програмний інтерфейс застосунку

DB - Database, база даних

ESP32 - мікроконтролер із вбудованим модулем бездротового зв’язку Wi-Fi

GPIO - General Purpose Input/Output, універсальні входи/виходи

HTTP - HyperText Transfer Protocol, протокол передавання гіпертексту

IDE - Integrated Development Environment, інтегроване середовище розробки

IoT - Internet of Things, Інтернет речей

JSON - JavaScript Object Notation, формат подання структурованих даних

MQTT - Message Queuing Telemetry Transport, протокол обміну повідомленнями між пристроями

POST - метод HTTP для надсилання даних на сервер

PubSub - модель взаємодії «публікація-підписка»

RSSI - Received Signal Strength Indicator, показник рівня прийнятого сигналу

TCP/IP - Transmission Control Protocol / Internet Protocol, стек мережевих протоколів

URL - Uniform Resource Locator, уніфікований покажчик ресурсу

Wi-Fi - технологія бездротового локального зв’язку

Wokwi - онлайн-середовище моделювання електронних схем і мікроконтролерів

СЕС - сонячна електростанція

ВСТУП

Актуальність теми. Сонячні електростанції зараз використовують все частіше, як у приватних господарствах, так і на підприємствах. Причина зрозуміла - це можливість отримувати електроенергію з відновлюваного джерела та зменшувати залежність від традиційних способів енергопостачання. Але на практиці самої наявності сонячних панелей недостатньо. Щоб система працювала стабільно, потрібно бачити, в якому стані знаходяться окремі панелі та як змінюються їхні робочі параметри.

Під час роботи СЕС на ефективність панелей впливають не тільки погодні умови. Значення мають також температура поверхні, рівень освітленості, можливе затінення, забруднення та окремі технічні несправності. Якщо контролювати лише загальні показники всієї станції, частину проблем можна виявити занадто пізно. Через це більш корисним є моніторинг не тільки всієї системи в цілому, а й окремих вузлів.

Для вирішення такої задачі доцільно використовувати мікроконтролерні пристрої, які можуть збирати дані з датчиків і передавати їх бездротовим способом. У даній роботі для цього обрано ESP32, оскільки цей мікроконтролер має вбудований модуль Wi-Fi, достатню кількість інтерфейсів і добре підходить для задач, пов'язаних з IoT. Для обміну даними між вузлами системи та контролером-шлюзом використовується MQTT, а для подальшої передачі інформації на сервер - HTTP.

Окремою перевагою такого підходу є можливість перевірити роботу системи без використання реального обладнання. Для цього можна застосувати середовище Wokwi, у якому зручно моделювати мікроконтролерні схеми, тестувати логіку програми та перевіряти обмін даними між елементами системи. Це особливо корисно на етапі проєктування, коли потрібно спочатку переконатися, що загальна структура системи працює правильно.

Отже, тема дипломної роботи є актуальною, оскільки пов'язана з розробкою доступної бездротової системи, яка дозволяє контролювати стан сонячних панелей, своєчасно виявляти відхилення в їх роботі та передавати зібрані дані для подальшої обробки.

Метою дипломної роботи є розробка бездротової мережі для моніторингу панелей у сонячній електростанції з використанням мікроконтролерних вузлів збору даних, контролера-шлюзу та засобів передавання інформації на сервер.

Для досягнення поставленої мети в роботі потрібно вирішити такі задачі:

1. виконати аналіз предметної області та розглянути особливості сонячних електростанцій як об'єкта моніторингу;
2. визначити основні параметри, які потрібно контролювати під час роботи сонячних панелей;
3. обрати апаратні та програмні засоби для побудови системи;
4. розробити структуру вузла моніторингу сонячної панелі;
5. розробити структуру контролера-шлюзу для приймання та обробки даних;
6. реалізувати зчитування температури, освітленості, напруги та струму;
7. реалізувати передавання телеметричних даних у форматі JSON через MQTT;
8. організувати передавання інформації від шлюзу на сервер через HTTP;
9. виконати моделювання та перевірку працездатності системи в середовищі Wokwi.

Об'єктом дослідження є процес бездротового моніторингу параметрів сонячних панелей у складі сонячної електростанції.

Предметом дослідження є апаратні та програмні засоби побудови багатовузлової системи збору, обробки та передачі даних про стан панелей СЕС.

Практичне значення одержаних результатів полягає в розробці моделі системи моніторингу, яка дає змогу зчитувати температуру, освітленість, напругу, струм і потужність окремих панелей, передавати ці дані до контролера-шлюзу, виконувати їх базову обробку та надсилати результати на сервер. Розроблене рішення може бути основою для подальшого розширення системи та її адаптації до використання з реальним обладнанням.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Особливості сонячних електростанцій як об’єкта моніторингу

Сонячна електростанція є об’єктом, для якого моніторинг має важливе практичне значення, оскільки ефективність її роботи залежить як від технічного стану обладнання, так і від зовнішніх умов експлуатації [8]. Її робота залежить не тільки від самого обладнання, а й від зовнішніх умов, які постійно змінюються. На ефективність генерації впливають освітленість, температура панелей, погодні умови, затінення, а також стан поверхні фотоелектричних модулів.

Основу СЕС складають сонячні панелі, які об’єднуються в модулі, стрінги та більші групи. Через це система працює не ізольовано, а як пов’язаний між собою комплекс. Якщо погіршуються параметри хоча б одного елемента, це може впливати не лише на нього, а й на роботу всієї частини станції. Наприклад, перегрів окремої панелі, часткове затінення або погіршення електричних характеристик можуть викликати зниження загальної потужності.

Ще одна особливість СЕС полягає в тому, що її параметри не залишаються сталими навіть протягом одного дня, оскільки рівень генерації залежить від освітленості, температури та поточних погодних умов [8]. Освітленість змінюється залежно від часу доби й погоди, температура панелей також постійно коливається. Разом із цим змінюються напруга, струм і потужність. Тому разового вимірювання тут недостатньо. Щоб реально оцінити стан системи, потрібно збирати дані регулярно та бачити, як саме вона поводиться в часі.

Окрему складність створює те, що сонячна електростанція має розподілену структуру. Навіть у невеликих системах використовується кілька панелей, а у більших - їх уже значно більше. У такій ситуації контроль лише загальних параметрів станції не завжди дозволяє швидко зрозуміти причину падіння продуктивності. Наприклад, зменшення загальної генерації може бути наслідком як зміни погоди, так і несправності однієї конкретної панелі. Без детальнішого контролю це визначити складно.

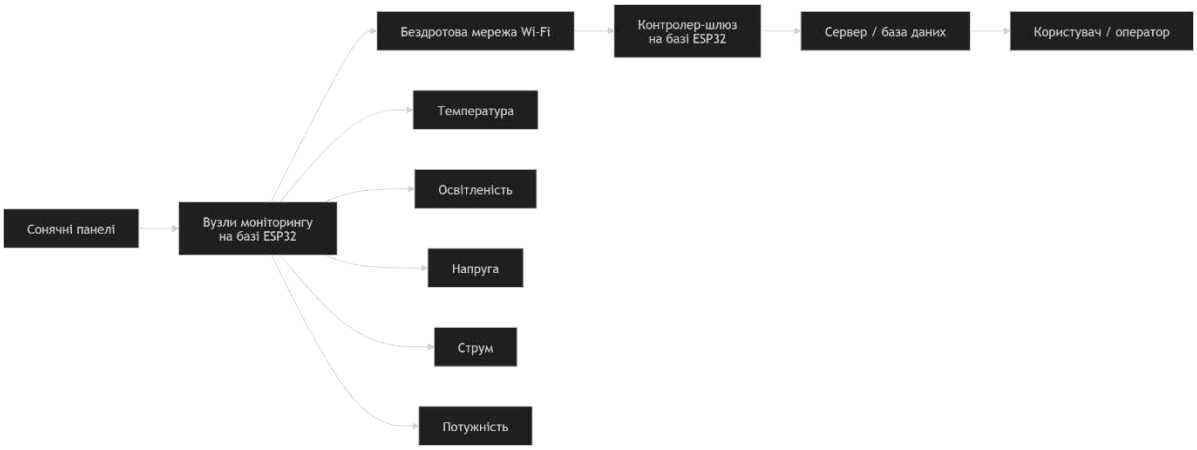

Рисунок 1.1 - Загальна структура системи моніторингу панелей у СЕС

На рисунку 1.1 показано загальну структуру системи моніторингу панелей у СЕС. Вузли моніторингу, побудовані на базі ESP32, збирають основні параметри роботи сонячних панелей, після чого передають їх через бездротову мережу Wi-Fi до контролера-шлюзу. Далі оброблена інформація надсилається на сервер, де може зберігатися, аналізуватися та відображатися для користувача.

Для практичного моніторингу СЕС доцільно відстежувати температуру, освітленість, напругу, струм і потужність. Саме ці параметри дають можливість оцінити режим роботи панелей, побачити відхилення та вчасно реагувати на проблеми. Тому система моніторингу для сонячної електростанції є не просто додатковою функцією, а важливою частиною її нормальної роботи.

1.2 Основні параметри, що підлягають контролю в роботі сонячних панелей

Щоб оцінити стан сонячної панелі під час роботи, потрібно контролювати кілька основних параметрів. Вони показують як умови, у яких працює панель, так і фактичну ефективність її генерації. На основі цих даних можна зрозуміти, чи працює панель нормально, і вчасно помітити небажані зміни.

Коли температура зростає, робоча напруга сонячного модуля зменшується, а разом із нею знижується і вихідна потужність [10]. Крім того, перегрів може бути ознакою не тільки сильного нагрівання від сонця, а й певних локальних проблем у модулі.

Не менш важливим є контроль освітленості. Цей параметр показує, скільки сонячної енергії потрапляє на поверхню панелі, а отже безпосередньо впливає на рівень генерації. Якщо освітленість висока, панель працює ближче до свого нормального

режиму. Якщо вона зменшується, потужність теж падає. Саме тому значення освітленості потрібне не саме по собі, а ще й для правильного тлумачення інших параметрів.

Серед електричних характеристик основними є напруга та струм. Вони відображають реальний режим генерації. Напруга показує рівень електричного потенціалу на виході панелі й залежить від температури, стану модуля та режиму навантаження. Струм, у свою чергу, тісно пов'язаний з освітленістю та показує інтенсивність передавання електричної енергії. Аналіз цих двох величин дозволяє оцінити, як саме працює панель у конкретний момент.

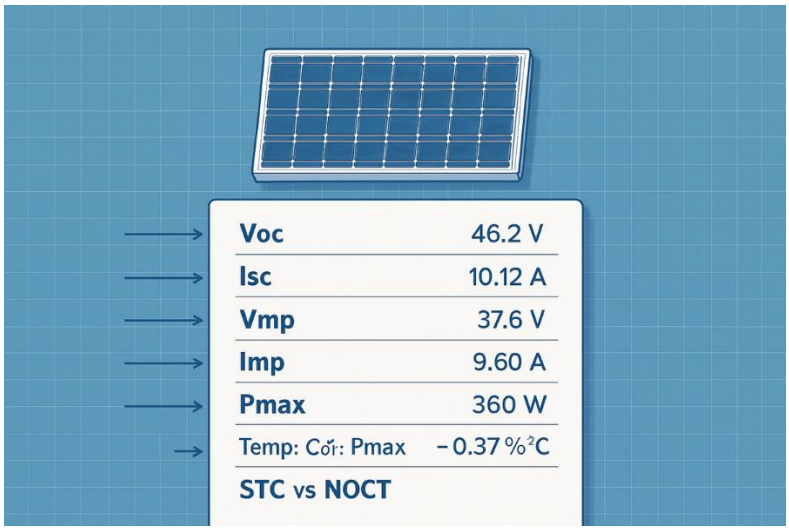

Рисунок 1.2 - Основні електричні характеристики сонячної панелі

На рисунку 1.2 наведено приклад основних електричних характеристик сонячної панелі, які використовуються для оцінки її робочого режиму. Значення напруги, струму та максимальної потужності дозволяють визначити ефективність генерації та є важливими параметрами під час аналізу роботи панелі.

На основі напруги та струму визначають потужність. Саме вона найкраще показує практичний результат роботи панелі. Цей параметр зручний тим, що за ним можна швидко порівнювати панелі між собою та оцінювати ефективність усієї системи в цілому.

У реальних системах інколи додатково враховують і інші показники, наприклад стабільність зв'язку, час останньої передачі даних або службовий стан вузла. Але для цієї роботи основну увагу доцільно зосередити саме на температурі, освітленості,

напрузі, струмі та потужності. Цього достатньо, щоб контролювати стан сонячної панелі та виявляти основні відхилення в її роботі.

1.3 Значення моніторингу в системах сонячної генерації

Для оцінки роботи сонячної електростанції недостатньо дивитися лише на загальний виробіток електроенергії. Такий показник дає тільки загальну картину, але не пояснює, що саме відбувається з окремими панелями та як змінюються їхні параметри під час експлуатації. Саме тому моніторинг у системах сонячної генерації має важливе практичне значення.

На ефективність роботи СЕС впливає багато чинників. Насамперед це освітленість і температура панелей. Крім них, значення мають погодні умови, часткове затінення, забруднення поверхні модулів і можливі технічні несправності. Навіть якщо проблема виникла лише в одній панелі, це все одно може позначитися на роботі цілої ділянки системи.

Система моніторингу дозволяє регулярно отримувати дані про температуру, освітленість, напругу, струм і потужність окремих панелей, що дає можливість оцінювати їхній технічний стан і своєчасно виявляти відхилення в роботі [7]. На основі цих даних можна оцінити, наскільки стабільно працює система, а також вчасно побачити відхилення від нормального режиму. Наприклад, якщо потужність знижується при нормальній освітленості, це вже може свідчити про перегрів, несправність або іншу проблему.

Ще одна важлива перевага моніторингу полягає в тому, що він дає не разове значення, а зміну параметрів у часі. Це особливо важливо для розподілених систем, де, як зазначено в джерелі, «використовується ряд протоколів, необхідних для обміну даними між вузлами, маршрутизаторами і хмарними сервісами в межах IoT-системи» [7]. Завдяки цьому можна порівнювати роботу панелей у різні періоди доби, відстежувати динаміку та бачити, як система поводиться за однакових або схожих умов.

На практиці це означає, що моніторинг дає змогу не просто збирати числа, а реально оцінювати технічний стан панелей. Через це проблемні місця можна виявити швидше, а рішення щодо експлуатації системи приймати точніше.

1.4 Аналіз сучасних систем моніторингу сонячних електростанцій

У сучасних системах моніторингу сонячних електростанцій зазвичай використовують два підходи. Перший - це готові комерційні платформи. Другий - створення власного рішення на базі мікроконтролерів. Вибір між ними залежить від вартості, гнучкості налаштування та того, що саме потрібно отримати від системи.

Комерційні системи моніторингу зазвичай розраховані на роботу з обладнанням певного виробника або з конкретними інверторами. Їхня перевага в тому, що користувач одразу отримує налаштовані засоби збору, передачі та відображення даних. Зазвичай такі рішення дозволяють переглядати стан системи через вебінтерфейс або мобільний застосунок. Це зручно, але при цьому подібні системи часто обмежують можливість змінювати структуру, підключати додаткові датчики або адаптувати роботу під власну задачу.

Для навчального проєктування цього не завжди достатньо. У межах бакалаврської роботи важливо не лише показати кінцевий результат моніторингу, а й продемонструвати принцип побудови системи, логіку взаємодії між її елементами, спосіб збирання та обробки даних. Якщо використовувати повністю готову платформу, значна частина таких рішень уже реалізована виробником і фактично залишається поза межами власної розробки. Через це можливості самостійного проєктування та обґрунтування прийнятих технічних рішень стають більш обмеженими.

Інший варіант - побудувати власну систему на базі мікроконтролерної платформи, зокрема ESP32, і перевірити її роботу в середовищі моделювання. Такий підхід є зручним, оскільки Wokwi підтримує моделювання проєктів на базі ESP32 [9]. У такому випадку можна самостійно визначити склад обладнання, набір контрольованих параметрів, спосіб передавання даних і загальну логіку роботи системи. Цей підхід потребує окремого проєктування і програмної реалізації, зате дає набагато більше свободи.

Крім цього, власна розробка дає змогу самостійно визначати склад вузлів моніторингу, набір контрольованих параметрів, формат повідомлень і порядок передавання даних між панелями, шлюзом та сервером. Для дипломної роботи це є важливим, оскільки дозволяє показати не лише використання готових інструментів, а

саме процес побудови системи на рівні архітектури, програмної логіки та взаємодії окремих компонентів.

Якщо порівнювати готові комерційні рішення, дротові схеми передавання даних і власну бездротову систему на базі мікроконтролерів, то краще саме третій варіант. Дротове підключення ускладнює побудову багатовузлової системи та робить її менш гнучкою при подальшому розширенні. Комерційні платформи є зручними в експлуатації, однак вони значно менше підходять для навчального проєктування, де потрібно показати логіку побудови системи та обґрунтувати власні технічні рішення. Натомість використання ESP32 дає можливість поєднати бездротовий обмін даними, гнучкість налаштування та доступність реалізації в межах навчальної моделі.

Для цієї роботи доцільніше обрати другий підхід. Він дозволяє не просто використати готовий інструмент, а самостійно побудувати структуру системи, перевірити взаємодію між її елементами й відпрацювати логіку обміну даними.

Додатковою перевагою є те, що ESP32 разом із Wokwi дозволяє створити модель системи без використання реального обладнання на початковому етапі. Це зручно для перевірки програми, тестування зв’язку між вузлами та подальшого розширення системи.

Вибір такого підходу обумовлений кількома критеріями. До них належать гнучкість налаштування системи, доступність апаратної платформи, зручність моделювання, можливість реалізації бездротового обміну даними без додаткового ускладнення схеми, а також придатність системи до подальшого масштабування. Саме сукупність цих факторів робить власну розробку більш доцільною для виконання дипломної роботи. Тому найбільш придатним є створення власної системи моніторингу на базі ESP32 з використанням Wi-Fi, MQTT і моделюванням у середовищі Wokwi.

1.5 Аналіз бездротових технологій передавання даних

Під час розробки системи моніторингу одним із головних питань є вибір технології бездротового зв’язку. Від цього залежить спосіб передавання даних між вузлами, швидкість обміну, складність реалізації та можливість подальшого розширення системи.

Одним із найпоширеніших варіантів є Wi-Fi. Його перевагами є достатньо висока швидкість передавання даних, широка доступність, підтримка великою кількістю мікроконтролерів і зручне підключення до локальної мережі та серверної частини [1]. Для систем, де передаються невеликі пакети телеметрії, можливостей Wi-Fi цілком достатньо.

Іншим відомим рішенням є Bluetooth Low Energy. Ця технологія орієнтована на енергоощадний обмін даними на невеликих відстанях, але є менш зручною для побудови багатовузлових централізованих систем моніторингу [2]. Тому BLE частіше застосовується в персональних або мобільних пристроях, а не в системах моніторингу сонячних електростанцій.

У промислових і спеціалізованих рішеннях також використовуються ZigBee та LoRa. ZigBee застосовується для побудови енергоефективних мереж із великою кількістю вузлів, тоді як LoRa орієнтована на передавання даних на великі відстані за невисокої швидкості обміну [3], [4].

Для даної системи найбільш доцільно використати саме Wi-Fi. Такий вибір пояснюється тим, що мікроконтролер ESP32 має вбудовану підтримку Wi-Fi і Bluetooth, а його можливостей достатньо для стабільної передачі телеметричних даних у межах розробленої системи [1]. Крім того, Wi-Fi зручно поєднується з MQTT, HTTP та добре підтримується в середовищі Wokwi.

У межах цієї роботи використання Wi-Fi дозволяє реалізувати зв’язок між вузлами панелей і контролером-шлюзом без застосування додаткових радіомодулів. Це спрощує і апаратну, і програмну частину системи.

1.6 Аналіз протоколів обміну даними в IoT-системах

Окрім вибору бездротової технології, для системи моніторингу потрібно визначити і протоколи обміну даними між її елементами. У сучасних IoT-рішеннях найчастіше використовуються HTTP, MQTT та інші полегшені протоколи, які відрізняються принципом роботи та сферою застосування.

HTTP є універсальним мережевим протоколом, який широко застосовується для взаємодії із серверами, вебсервісами та API, оскільки працює за клієнт-серверною моделлю обміну даними [5]. Його перевага полягає в тому, що через нього зручно

передавати вже підготовлені дані, зберігати їх у базі або використовувати у вебінтерфейсі. Для таких задач цей протокол є зрозумілим і досить простим у реалізації.

Разом із тим HTTP не завжди зручний для прямого обміну телеметрією між великою кількістю мікроконтролерних вузлів. Причина в тому, що HTTP працює за схемою "запит - відповідь", у якій клієнт надсилає запит, а сервер формує окрему відповідь, що не завжди є зручним для регулярного обміну короткими телеметричними повідомленнями [5]. Для пристроїв з обмеженими ресурсами та регулярною передачею невеликих пакетів даних це не завжди оптимально.

MQTT, навпаки, спеціально орієнтований на IoT-пристрої та системи телеметрії. Він працює за принципом "публікація - підписка" і належить до полегшених протоколів передавання повідомлень, зручних для пристроїв з обмеженими ресурсами [6]. При цьому одні вузли надсилають повідомлення в певні канали, а інші отримують їх після підписки на відповідні топіки. Завдяки цьому MQTT добре підходить для багатовузлових систем, у яких потрібно організувати зручний обмін даними між кількома джерелами та центральним вузлом.

До переваг MQTT належать невеликий службовий трафік, компактність повідомлень, зручність масштабування та хороша підтримка на мікроконтролерних платформах, зокрема на ESP32, що робить його придатним для телеметричних IoT-систем [6]. Саме тому цей протокол доцільно використовувати для внутрішнього обміну телеметрією між вузлами панелей і контролером-шлюзом.

У даній роботі використано комбінований підхід: між панелями та шлюзом застосовується MQTT, а між шлюзом і сервером - HTTP. Така схема дозволяє поєднати сильні сторони обох протоколів. MQTT зручний для внутрішнього обміну даними всередині системи, а HTTP добре підходить для подальшого передавання обробленої інформації на сервер.

1.7 Постановка задачі дипломної роботи

Проведений аналіз показав, що для ефективної роботи сонячної електростанції бажано контролювати параметри окремих панелей, а не обмежуватися лише загальними показниками всієї системи. Це дає можливість точніше оцінювати фактичний стан обладнання та швидше виявляти відхилення в роботі окремих вузлів.

Готові комерційні рішення не завжди зручні для навчального проєктування. Вони часто мають обмеження щодо зміни структури системи, підключення додаткових датчиків і реалізації власної логіки обробки даних. Крім того, використання готових платформ не завжди дозволяє повністю простежити всі етапи побудови системи - від отримання даних із датчиків до передачі інформації на сервер.

З урахуванням цього в межах дипломної роботи поставлено задачу розробити бездротову систему моніторингу панелей у СЕС на базі ESP32. Система повинна забезпечувати зчитування температури, освітленості, напруги та струму, обчислення потужності панелі, передавання даних до контролера-шлюзу та подальше надсилання інформації на сервер.

Для реалізації поставленої задачі необхідно:

1. обрати апаратні та програмні засоби реалізації системи;
2. розробити структуру вузла моніторингу сонячної панелі;
3. розробити структуру контролера-шлюзу;
4. реалізувати бездротовий обмін даними між вузлами системи;
5. організувати передавання інформації на сервер;
6. виконати моделювання та перевірку працездатності системи в середовищі Wokwi.

У результаті має бути створена функціональна модель бездротової системи моніторингу, яка дозволяє збирати, передавати та обробляти основні параметри роботи сонячних панелей.

1.8 Висновки до розділу 1

У першому розділі було розглянуто особливості сонячних електростанцій як об’єкта моніторингу та визначено основні параметри, які доцільно контролювати під час їхньої роботи. Показано, що для оцінки стану панелей найбільше значення мають температура, освітленість, напруга, струм і потужність, оскільки саме ці показники відображають поточний режим генерації та дозволяють виявляти відхилення.

Проведений аналіз також показав, що контроль лише загальних параметрів станції не завжди дає змогу своєчасно помітити локальні проблеми окремих панелей.

Через це більш доцільним є підхід, за якого дані збираються з окремих вузлів моніторингу та передаються до центрального елемента системи.

Під час аналізу сучасних рішень було встановлено, що для цієї роботи більш придатним є створення власної системи моніторингу на базі мікроконтролерної платформи ESP32. Такий підхід дозволяє самостійно визначити структуру системи, набір контрольованих параметрів і логіку взаємодії між її елементами.

Окремо були розглянуті бездротові технології та протоколи обміну даними. Для побудови системи обрано Wi-Fi як засіб зв’язку між вузлами, MQTT - для передавання телеметрії між панелями та шлюзом, а HTTP - для надсилання даних на сервер. Така комбінація відповідає поставленій задачі та є зручною для реалізації в середовищі Wokwi.

На основі проведеного аналізу сформульовано задачу дипломної роботи, яка полягає у розробці бездротової системи моніторингу панелей у СЕС із використанням вузлів збору даних, контролера-шлюзу та серверної частини.

РОЗДІЛ 2 ПРОЄКТУВАННЯ БЕЗДРОТОВОЇ СИСТЕМИ МОНІТОРИНГУ ПАНЕЛЕЙ У СЕС

2.1 Загальна структура системи моніторингу

Під час проєктування системи моніторингу панелей сонячної електростанції потрібно було побудувати таку структуру, яка б дозволяла збирати дані з кількох вузлів, передавати їх бездротовою мережею, обробляти в одному центральному елементі та далі надсилати на сервер. При цьому система мала залишатися достатньо простою, щоб її можна було реалізувати та перевірити в середовищі Wokwi. Основою для побудови системи стала трирівнева структура, що включає вузли моніторингу панелей, контролер-шлюз та серверну частину.

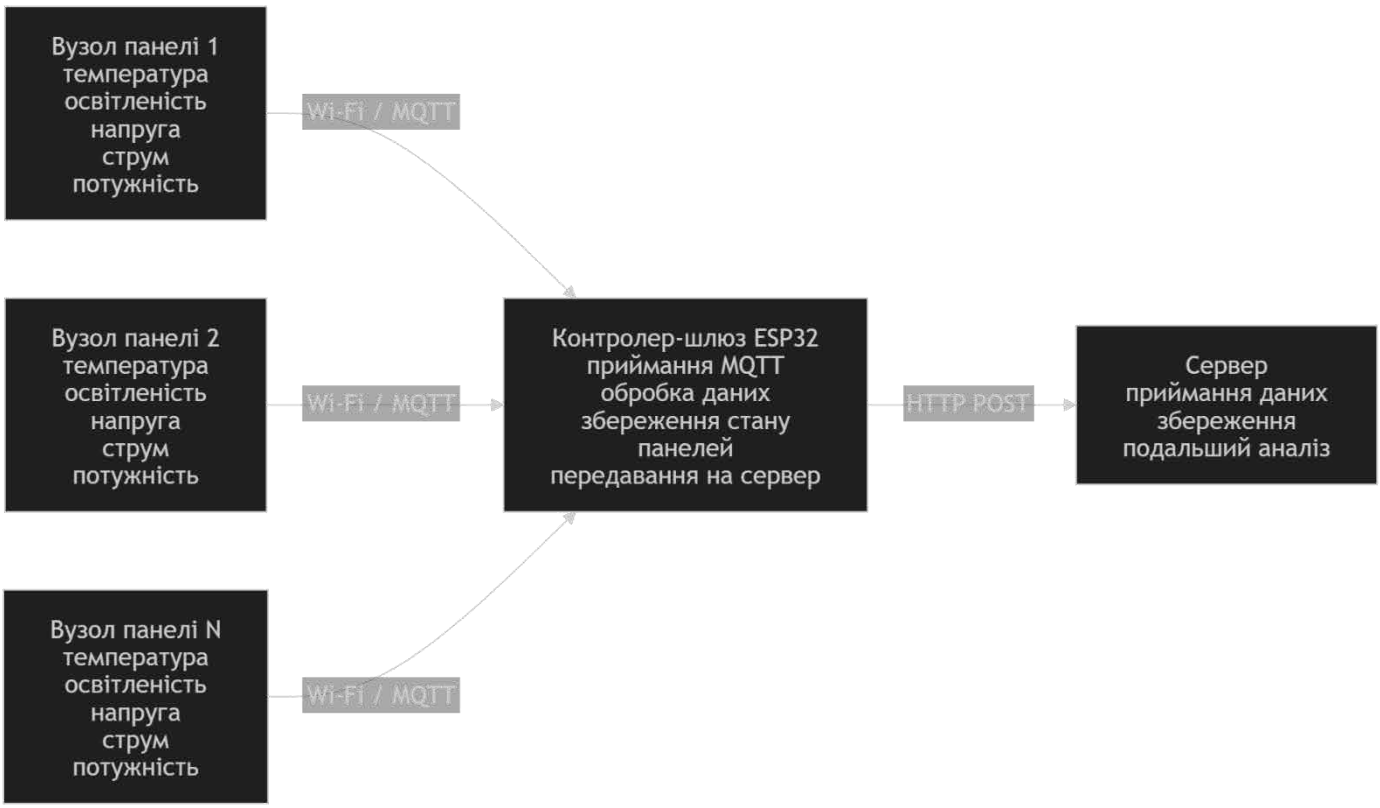

Рисунок 2.1 - Структурна схема системи моніторингу панелей у СЕС

Розроблена система складається з трьох основних частин:

- вузли моніторингу сонячних панелей;
- контролер-шлюз збору та обробки даних;
- сервер для приймання інформації.

Кожен вузол моніторингу працює з окремою сонячною панеллю. Його задача полягає у зчитуванні основних параметрів, первинній обробці отриманих значень, формуванні повідомлення у форматі JSON та передаванні цього повідомлення в мережу. Кожен такий вузол має власний ідентифікатор, тому дані від різних панелей можна без проблем розділяти під час подальшої обробки. Такий підхід забезпечує зручне розділення даних від різних панелей під час подальшої обробки та аналізу.

Центральним елементом системи є контролер-шлюз. Саме він приймає повідомлення від вузлів, визначає, від якої панелі надійшли дані, зберігає поточний стан кожного вузла та виконує базову обробку отриманої інформації. Після цього дані передаються на сервер для подальшого перегляду, збереження або аналізу.

Серверна частина у даній роботі використовується як точка приймання даних від шлюзу. На цьому рівні інформація може накопичуватися, журналюватися або використовуватися для подальшої візуалізації. У межах моделі сервер не ускладнювався зайвими функціями, оскільки основна увага приділялася саме побудові каналу передавання даних від панельного вузла до шлюзу і далі на сервер.

Загальна логіка роботи системи є такою. Вузол панелі періодично опитує датчики, після чого формує структуроване повідомлення та передає його через MQTT. Контролер-шлюз приймає це повідомлення, оновлює інформацію про відповідну панель, а потім надсилає оброблені дані на сервер за допомогою HTTP POST. Така схема дозволяє реалізувати централізований моніторинг без складної кабельної інфраструктури й добре підходить для багатовузлової системи. Саме така послідовність роботи покладена в основу подальшої програмної реалізації системи.

Перевага такої структури полягає в тому, що вона охоплює всі основні етапи роботи системи: зчитування даних, їх бездротове передавання, централізоване приймання, базову обробку та подальше надсилання на сервер. При цьому структура не є надто складною, тому її зручно реалізовувати в середовищі Wokwi та використовувати як основу для подальшого розширення системи.

2.2 Архітектура мережі вузлів панелей і контролера-шлюзу

Архітектура розробленої системи побудована як багатовузлова бездротова мережа з одним центральним шлюзом. Такий підхід обрано для того, щоб дані від

кількох панелей можна було збирати в одному місці, обробляти та далі передавати на сервер. У цій схемі панельні вузли не обмінюються інформацією між собою напряму, а працюють через контролер-шлюз. Саме він виступає центральним елементом усієї системи.

У межах цієї роботи основна модель мережі побудована за централізованою схемою, де кожен вузол панелі передає дані безпосередньо до контролера-шлюзу. Такий варіант був обраний як базовий, оскільки він простіший для моделювання у Wokwi, зручний для перевірки логіки обміну даними та дозволяє наочно показати роботу всієї системи. Разом із цим сама архітектура не обмежується лише такою схемою. При подальшому розширенні її можна адаптувати до mesh-підходу, коли окремі вузли можуть не тільки передавати власні дані, а й виконувати роль проміжних учасників мережі для ретрансляції повідомлень.

Використання mesh-логіки є доцільним для більших сонячних електростанцій, де панелі можуть бути розміщені на значній площі або частина вузлів може перебувати поза стабільною зоною зв’язку зі шлюзом. У такому випадку дані від віддалених вузлів можуть передаватися до контролера-шлюзу не напряму, а через сусідні вузли. Це підвищує гнучкість мережі та зменшує залежність усієї системи від одного прямого каналу зв’язку. У поточній реалізації було перевірено базову централізовану взаємодію, а mesh-структура розглядається як напрям подальшого масштабування системи.

Кожен вузол панелі підключається до бездротової мережі Wi-Fi і виконує роль джерела телеметричних даних. Після зчитування параметрів вузол формує повідомлення у форматі JSON та передає його через MQTT. Для того щоб шлюз міг розрізняти джерела інформації, для кожної панелі використовується окремий ідентифікатор. За рахунок цього можна точно визначити, від якого саме вузла надійшли дані, і правильно зберігати їх у системі.

Контролер-шлюз у цій архітектурі виконує одразу кілька задач. Він приймає повідомлення від усіх панельних вузлів, розбирає отримані дані, оновлює поточний стан кожної панелі та передає зібрану інформацію далі на сервер. Така організація зручна тим, що вся логіка обробки зосереджується в одному місці. Через це простіше

контролювати роботу системи, додавати нові вузли та вносити зміни в програму шлюзу без переробки всієї мережі.

Ще одна перевага такої архітектури полягає в її масштабованості. Якщо кількість панелей збільшується, до системи можна додати нові вузли моніторингу без зміни її базового принципу роботи. При цьому загальна структура мережі залишається тією самою: панельні вузли збирають і надсилають дані, а шлюз приймає їх і виконує централізовану обробку. Для навчальної моделі це важливо, оскільки система має бути не тільки працездатною, а й придатною до подальшого розширення.

Отже, вибрана архітектура мережі відповідає поставленій задачі дипломної роботи. Вона дозволяє організувати збір даних від кількох панелей, забезпечити їх бездротове передавання, централізовану обробку на шлюзі та подальше надсилання на сервер. Для моделювання в середовищі Wokwi та для подальшої практичної реалізації така схема є зручною і цілком виправданою.

2.3 Проєктування вузла моніторингу сонячної панелі

Вузол моніторингу сонячної панелі є окремим мікроконтролерним пристроєм, призначеним для локального збору телеметричних даних. Його основне завдання полягає у зчитуванні параметрів панелі, первинній обробці отриманої інформації та подальшому передаванні даних до контролера-шлюзу. Для реалізації такого вузла необхідно, щоб він підтримував роботу як із цифровими, так і з аналоговими датчиками, а також мав можливість бездротового обміну даними.

У межах цієї роботи вузол моніторингу повинен виконувати кілька основних функцій:

- підключення до бездротової мережі;
- ініціалізацію датчиків;
- періодичне зчитування температури;
- зчитування значень напруги, струму та освітленості;
- обчислення фактичних параметрів роботи панелі;
- формування JSON-повідомлення;
- передавання даних через MQTT.

Апаратною основою вузла обрано мікроконтролер ESP32. Це рішення пов’язане з тим, що ESP32 має вбудований модуль Wi-Fi, достатню кількість входів і виходів, а також добре підтримується в середовищі Wokwi. Для вимірювання температури використовується датчик DS18B20, а для моделювання напруги, струму та освітленості застосовуються аналогові входи мікроконтролера.

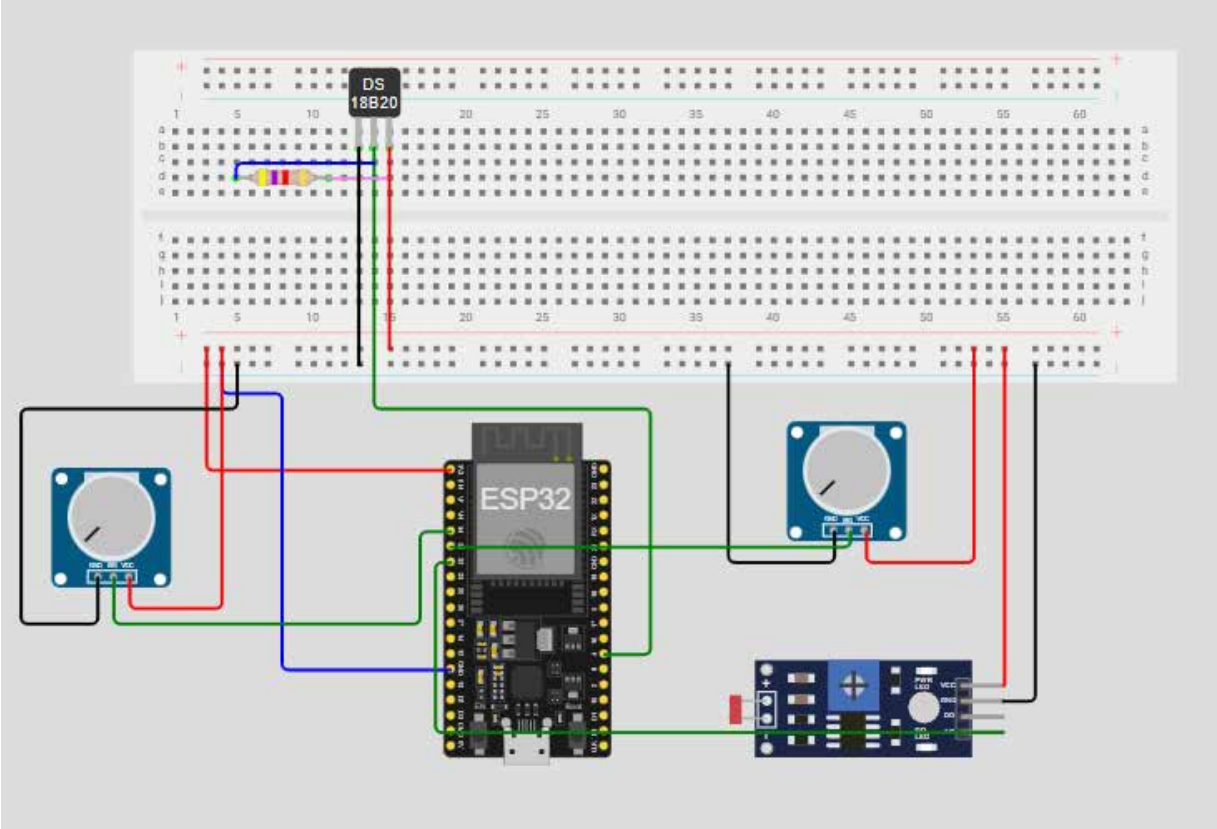

Рисунок 2.2 - Апаратна основа вузла моніторингу сонячної панелі на базі ESP32

На рисунку 2.2 показано апаратну основу вузла моніторингу, реалізовану на базі ESP32 у середовищі Wokwi. Саме цей мікроконтролер забезпечує зчитування температури, освітленості, напруги та струму, подальшу обробку отриманих значень, формування JSON-повідомлення і передавання даних через бездротову мережу до контролера-шлюзу.

Для розрізнення даних від різних панелей у вузлі використовується індивідуальний ідентифікатор. Він додається до повідомлення та використовується в назві MQTT-топіка. Це дозволяє шлюзу визначати, від якої саме панелі надійшли дані, і спрощує подальше розширення системи.

Періодичність надсилання телеметрії обрана так, щоб система регулярно оновлювала інформацію, але не створювала зайвого навантаження на бездротову

мережу. У поточній моделі цього достатньо для демонстрації роботи вузла в умовах моделювання.

У результаті вузол моніторингу реалізовано як автономний модуль збору та передавання телеметричних даних, який може працювати в складі багатовузлової системи моніторингу сонячної електростанції.

2.4 Проєктування контролера-шлюзу

Контролер-шлюз є центральним елементом розробленої системи моніторингу. Його призначення полягає у прийманні повідомлень від вузлів панелей, збереженні поточного стану системи, базовій обробці отриманих даних та подальшому передаванні інформації на сервер. На відміну від вузлів панелей, які працюють переважно як джерела телеметрії, шлюз виконує координуючу функцію і об’єднує всі елементи системи в єдину структуру.

Апаратною основою контролера-шлюзу обрано мікроконтролер ESP32. Такий вибір пояснюється наявністю вбудованого модуля Wi-Fi, достатньою обчислювальною продуктивністю, а також підтримкою MQTT, HTTP і засобів роботи з JSON-даними. Завдяки цьому ESP32 може використовуватися не лише як вузол збору даних, а й як центральний пристрій, який приймає телеметрію від панелей, аналізує її та передає результати на сервер.

У межах цієї роботи контролер-шлюз виконує такі основні функції:

- підключення до бездротової мережі Wi-Fi;
- підключення до MQTT-брокера;
- приймання повідомлень від вузлів панелей;
- розбір JSON-повідомлень;
- збереження параметрів кожної панелі у внутрішній структурі даних;
- передавання інформації на сервер через HTTP;
- аналіз поточного стану генерації;
- виведення інформаційних та попереджувальних повідомлень.

Рисунок 2.3 - Апаратна основа контролера-шлюзу на базі ESP32

На рисунку 2.3 показано апаратну основу контролера-шлюзу, реалізовану на базі ESP32. У даній системі цей мікроконтролер використовується як центральний вузол, що приймає повідомлення від панелей, виконує їх обробку та надсилає дані до серверної частини.

Для зберігання поточного стану панелей у шлюзі використовується окрема структура даних, у якій містяться ідентифікатор панелі, температура, напруга, струм, потужність, рівень освітленості, код останньої відповіді сервера та ознака активності вузла. Такий підхід дозволяє зберігати актуальну інформацію про кожну панель окремо та працювати з нею вже на рівні шлюзу.

Окрім простого передавання даних, шлюз виконує і базовий аналіз стану системи. У поточній реалізації він обчислює сумарну потужність активних панелей і середній рівень освітленості. На основі цих значень формується оцінка поточного режиму роботи. Якщо сумарна потужність перевищує допустимий рівень, система видає попередження про можливе перевантаження. Якщо генерація є недостатньою за низької освітленості, формується повідомлення про зниження ефективності роботи.

Таким чином, контролер-шлюз у цій системі виконує не лише роль проміжної ланки між панелями та сервером, а й забезпечує централізовану обробку та первинний

аналіз телеметричних даних. Це підвищує практичну цінність розробленої системи моніторингу та робить її більш функціональною.

2.5 Вибір апаратних компонентів системи

Вибір апаратних компонентів є важливим етапом проєктування, оскільки саме від нього залежать функціональні можливості системи, її вартість, надійність та зручність реалізації. Для даної дипломної роботи були обрані компоненти, які добре підтримуються у середовищі Wokwi, є поширеними серед розробників IoT-систем та дозволяють реалізувати поставлені задачі без значного ускладнення схеми.

Основою як вузлів панелей, так і шлюзу є мікроконтролер ESP32. Його вибір обумовлений такими причинами:

- наявність вбудованого модуля Wi-Fi;
- достатня кількість цифрових та аналогових входів/виходів;
- підтримка роботи з бібліотеками для MQTT, HTTP та датчиків;
- висока продуктивність порівняно з простішими мікроконтролерами;
- зручність програмування в Arduino IDE та моделювання у Wokwi.

Для вимірювання температури обрано цифровий датчик DS18B20, який використовує інтерфейс 1-Wire і добре підходить для мікроконтролерних систем моніторингу [11]. Його перевагами є:

- цифровий інтерфейс обміну;
- достатня точність вимірювання температури;
- простота підключення;
- широка підтримка у бібліотеках Arduino-сумісних платформ.

Для більш наочного подання апаратної конфігурації вузла моніторингу сонячної панелі доцільно подати підключення основних компонентів у вигляді таблиці. Це дозволяє чітко визначити відповідність між датчиками, допоміжними елементами та виводами мікроконтролера ESP32.

Таблиця 2.1 - Підключення компонентів вузла моніторингу сонячної панелі

Компонент	Призначення	Вивід компонента	Вивід ESP32
DS18B20	Вимірювання температури	DQ	GPIO 4
DS18B20	Живлення датчика	VCC	3.3V
DS18B20	Загальний провід	GND	GND
Резистор 4,7 кОм	Підтягування лінії OneWire	між VCC і DQ	між 3.3V та GPIO 4
Потенціометр 1	Імітація напруги	SIG	GPIO 34
Потенціометр 1	Живлення	VCC	3.3V
Потенціометр 1	Загальний провід	GND	GND
Потенціометр 2	Імітація струму	SIG	GPIO 35
Потенціометр 2	Живлення	VCC	3.3V
Потенціометр 2	Загальний провід	GND	GND
Фоторезисторний модуль	Вимірювання освітленості	AO	GPIO 32
Фоторезисторний модуль	Живлення	VCC	3.3V
Фоторезисторний модуль	Загальний провід	GND	GND

Як видно з таблиці 2.1, у розробленій системі використано комбіноване підключення цифрового датчика температури та аналогових джерел сигналів. Такий підхід дозволяє поєднати контроль температурного режиму панелі з моделюванням електричних параметрів і рівня освітленості у середовищі Wokwi.

Для визначення рівня освітленості, напруги та струму в моделі використовуються аналогові входи ESP32. У середовищі Wokwi ці значення можуть задаватися за допомогою потенціометрів або інших засобів моделювання, що дозволяє наочно демонструвати зміну параметрів системи в режимі реального часу. Такий підхід є зручним для навчального та дослідницького моделювання.

Оскільки система працює як багатовузлова бездротова мережа з централізованим шлюзом, однакові апаратні платформи для вузлів спрощують налаштування, підтримку та розширення системи. Це також полегшує адаптацію програмного коду, оскільки для різних панелей змінюється переважно ідентифікатор вузла, а основна логіка роботи залишається спільною.

2.6 Вибір програмних засобів розробки

Для реалізації системи моніторингу потрібно було обрати такі програмні засоби, які дозволяють зручно писати код, перевіряти його роботу та моделювати взаємодію

між елементами системи. У межах цієї дипломної роботи для цього використовуються Arduino IDE або сумісне середовище розробки, Wokwi, а також набір бібліотек для роботи з мережею, датчиками та структурованими даними.

Для написання програмного забезпечення під ESP32 доцільно використовувати Arduino IDE, оскільки це середовище є простим у використанні, широко застосовується під час розробки мікроконтролерних систем і має добру підтримку бібліотек для ESP32 [12]. Його перевагою є зрозумілий інтерфейс, велика кількість готових бібліотек, прикладів і хороша підтримка платформи ESP32.

Для моделювання роботи системи обрано середовище Wokwi. Воно дозволяє відтворити схему пристрою, підключення основних компонентів і перевірити логіку взаємодії між вузлами без використання фізичного обладнання. Для даної роботи це особливо зручно, оскільки дає можливість протестувати роботу вузлів панелей, контролера-шлюзу та обмін даними між ними в межах однієї моделі.

У програмній реалізації системи використовуються такі основні бібліотеки:

- **WiFi.h** - для підключення пристроїв до бездротової мережі;
- **PubSubClient.h** - для роботи з MQTT-брокером;
- **OneWire.h** та **DallasTemperature.h** - для взаємодії з датчиком температури DS18B20;
- **ArduinoJson.h** - для формування та обробки JSON-даних;
- **HTTPClient.h** - для надсилання HTTP-запитів на сервер.

Вибір саме цих бібліотек пояснюється тим, що вони є поширеними, стабільними та сумісними з ESP32. Крім того, їх функціональних можливостей достатньо для реалізації всіх основних задач цієї системи: підключення до Wi-Fi, обміну повідомленнями через MQTT, роботи з датчиком температури, формування JSON і передавання даних на сервер.

Таким чином, обрані програмні засоби дозволяють реалізувати повний цикл розробки та перевірки системи моніторингу - від написання коду до моделювання взаємодії між її елементами в середовищі Wokwi.

2.7 Вибір формату передавання та обробки даних

Для передавання телеметричних даних між елементами системи в роботі було обрано формат JSON. Такий вибір пов'язаний не тільки з його поширеністю, а й з тим, що він добре підходить саме для цієї реалізації на ESP32. JSON має просту структуру, легко формується в програмі мікроконтролера і без зайвих труднощів обробляється на стороні шлюзу та сервера. Формат зручний тим, що повідомлення легко сформувати на вузлі панелі й так само просто розібрати на стороні шлюзу.

У розробленій системі через JSON передаються основні параметри роботи панелі. До повідомлення входять ідентифікатор панелі, температура, напруга, струм, потужність і рівень освітленості. Такий набір полів дозволяє передати всю потрібну інформацію в одному повідомленні без ускладнення структури. Для шлюзу це зручно, оскільки він може швидко отримати значення, розкласти їх по відповідних змінних і далі використати для оновлення поточного стану системи. Саме такий набір полів є достатнім для передавання основних телеметричних параметрів панелі.

Перевага JSON у цьому випадку полягає ще й у тому, що формат легко розширюється. Якщо надалі виникне потреба додати часову мітку, службовий стан вузла, рівень сигналу або інші параметри, це можна зробити без повної переробки логіки обміну даними. Для навчальної моделі це теж важливо, бо система повинна залишатися зрозумілою, але при цьому мати можливість для подальшого розвитку.

Окремо важливо, що той самий формат використовується на кількох етапах передавання інформації. Спочатку дані формуються на вузлі панелі, далі приймаються і обробляються на шлюзі, після чого можуть бути передані на сервер. Такий підхід спрощує програмну реалізацію, оскільки не потрібно перетворювати повідомлення в інший формат на кожному етапі. Через це зменшується складність коду і полегшується перевірка роботи всієї системи.

Отже, вибір JSON для цієї системи є виправданим, оскільки він поєднує простоту, наочність і зручність практичного використання. Для розробленої моделі цього формату достатньо, щоб організувати передавання основних телеметричних даних між вузлами панелей, контролером-шлюзом і серверною частиною.

2.8 Розробка логіки взаємодії між елементами системи

Після визначення складу системи, способу бездротового зв’язку та формату повідомлень потрібно було побудувати зрозумілу логіку взаємодії між усіма її елементами. У розробленій системі логіка взаємодії побудована так, щоб кожен вузол виконував свою конкретну задачу, а обмін даними між панелями, шлюзом і сервером проходив послідовно та без зайвого ускладнення. Такий підхід відповідає загальній структурі системи з панельними вузлами, контролером-шлюзом і серверною частиною.

Початковою ланкою в системі є вузол моніторингу сонячної панелі. Він періодично зчитує значення температури, освітленості, напруги та струму, після чого на основі отриманих даних обчислює потужність панелі. Далі вузол формує JSON-повідомлення, у якому містяться основні параметри роботи та ідентифікатор конкретної панелі. Наявність окремого ідентифікатора дозволяє відразу визначити, від якого саме вузла надійшла інформація. У розробленій системі це також пов’язано з використанням MQTT-топіків і періодичним надсиланням телеметрії через заданий інтервал часу.

Після формування повідомлення дані передаються через бездротову мережу за допомогою MQTT. Такий спосіб обміну вибрано тому, що він добре підходить для багатовузлової системи, де кілька джерел даних передають інформацію до одного центрального елемента. Панельні вузли в цій схемі виступають як джерела телеметрії, а контролер-шлюз приймає всі повідомлення, що надходять від них. За рахунок цього логіка обміну залишається простою: вузли не взаємодіють напряму між собою, а передають дані через центральний вузол.

Контролер-шлюз у розробленій системі виконує роль основного координатора. Він підключається до Wi-Fi, встановлює з’єднання з MQTT-брокером, підписується на повідомлення від панелей і приймає їх у процесі роботи системи. Після отримання нового повідомлення шлюз виконує його розбір, витягує значення основних параметрів і зберігає їх у внутрішній структурі даних. Для кожної панелі підтримується окремий поточний стан, що дозволяє не просто приймати окремі пакети даних, а бачити актуальну картину по всіх вузлах системи. Для контролера-шлюзу визначено саме такі функції: приймання MQTT-повідомлень, розбір JSON-даних, збереження параметрів кожної панелі та передавання інформації на сервер.

Для зменшення ризику приймання сторонніх або помилкових повідомлень у системі передбачено просту перевірку автентичності вузла. Кожен вузол панелі додає до JSON-повідомлення службове поле `key`, у якому передається заздалегідь визначений ключ пристрою. Контролер-шлюз після отримання MQTT-повідомлення спочатку перевіряє наявність цього ключа та порівнює його з очікуваним значенням. Якщо ключ відсутній або не збігається, повідомлення відхиляється і не використовується для оновлення стану системи. Такий підхід не є повноцінним шифруванням, але в межах моделі дозволяє показати базовий захист від неавторизованої передачі телеметрії.

Після оновлення даних шлюз може виконувати дві основні дії. Перша - це передавання інформації на сервер через HTTP-запит. Друга - це локальний аналіз поточного стану системи, зокрема обчислення сумарної потужності активних панелей і перевірка умов, за яких виникає попередження про перевантаження або недостатню генерацію.

Крім звичайного приймання та передавання інформації, у роботі контролера-шлюзу передбачено елемент базового логічного аналізу стану системи. Після оновлення параметрів активних панелей шлюз обчислює сумарну потужність генерації та середній рівень освітленості. На основі цих двох показників він формує повідомлення про поточний режим роботи системи.

Якщо рівень освітленості є високим, а сумарна потужність перевищує встановлений поріг, шлюз виводить попередження про можливе перевантаження інвертора. У такому випадку система сигналізує, що генерація є надто великою для заданого режиму роботи, і рекомендує звернути увагу на навантаження або конфігурацію підключених панелей.

У випадку, коли освітленість знижується і одночасно сумарна потужність падає нижче заданого мінімального рівня, шлюз формує попередження про недостатню генерацію. Така ситуація може бути пов’язана з хмарністю, затіненням або несприятливими умовами роботи панелей. У цьому режимі система фактично переходить у стан зниженого виробітку та повідомляє про це користувача.

Якщо ж поточні параметри залишаються в допустимих межах, шлюз виводить повідомлення про нормальний або збалансований режим роботи. Таким чином, у

розробленій системі контролер-шлюз виконує не лише роль проміжного вузла передавання даних, а й функцію первинної оцінки стану генерації.

Наявність такої логіки робить систему більш інформативною, оскільки користувач отримує не тільки числові значення параметрів, а й короткий висновок щодо поточного стану роботи сонячної електростанції.

Для зручності контролю в системі також передбачено виведення табличного представлення поточних параметрів панелей у консольний інтерфейс. Це дозволяє візуально оцінювати стан системи під час моделювання та тестування.

Запропонована логіка взаємодії є достатньо простою, однак вона охоплює всі основні етапи роботи системи: зчитування, передавання, приймання, обробку, аналіз і збереження даних. Завдяки цьому розроблена система є цілісною та функціонально завершеною моделлю моніторингу панелей у СЕС.

2.9 Висновки до розділу 2

У другому розділі було розглянуто основні питання, пов’язані з проєктуванням бездротової системи моніторингу панелей у СЕС. Визначено загальну структуру системи, до складу якої входять вузли моніторингу сонячних панелей, контролер-шлюз та серверна частина для приймання й подальшої обробки даних.

У процесі проєктування було обґрунтовано архітектуру взаємодії між елементами системи. Показано, що використання окремих вузлів для кожної панелі та централізованого контролера-шлюзу дозволяє організувати зручний збір телеметрії, розділення даних за ідентифікаторами панелей і подальшу передачу інформації на сервер.

Окремо було розглянуто проєктування вузла моніторингу сонячної панелі та контролера-шлюзу. Для апаратної основи обох елементів обрано мікроконтролер ESP32, що пояснюється наявністю вбудованого модуля Wi-Fi, достатньою кількістю інтерфейсів і зручністю використання в середовищі Wokwi. Для вимірювання температури використано датчик DS18B20, а для моделювання напруги, струму та освітленості - аналогові входи мікроконтролера.

Також у розділі було обґрунтовано вибір програмних засобів розробки, зокрема Arduino IDE, середовища Wokwi та бібліотек WiFi, PubSubClient, OneWire,

DallasTemperature, ArduinoJson і HTTPClient. Для подання телеметричних даних обрано формат JSON, а для обміну між панелями та шлюзом використано MQTT. Передавання інформації від шлюзу на сервер реалізується через HTTP.

У результаті розроблено логіку взаємодії між усіма елементами системи, яка охоплює зчитування параметрів, формування повідомлень, бездротове передавання, приймання, обробку та базовий аналіз стану генерації. Таким чином, у другому розділі було сформовано повну проєктну основу для подальшої програмної реалізації системи.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Реалізація програмного забезпечення вузла сонячної панелі

Програмне забезпечення вузла сонячної панелі реалізовано на базі мікроконтролера ESP32. Основною задачею цього вузла є зчитування даних з датчиків, обчислення параметрів роботи панелі, формування структурованого повідомлення та його передавання до контролера-шлюзу через MQTT.

Для реалізації програми використовуються бібліотеки `WiFi.h`, `PubSubClient.h`, `OneWire.h` та `DallasTemperature.h`. Вони забезпечують роботу з мережею Wi-Fi, MQTT-протоколом і датчиком температури DS18B20.

На початку програми задаються основні параметри вузла моніторингу сонячної панелі. До них належать номери виводів, до яких підключені датчики та аналогові входи, параметри бездротової мережі, адреса MQTT-брокера, ідентифікатор вузла та назва топіка для передавання даних. Саме ці значення визначають базову конфігурацію панельного вузла та подальшу логіку його роботи.

Лістинг 3.1 - Оголошення параметрів вузла панелі

```
 #include <WiFi.h>
#include <PubSubClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>
const int oneWireBus = 4;
const int voltagePin = 34;
const int currentPin = 35;
const int lightPin = 32;
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);
const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";
const int PANEL_ID = 1;
const char* topic_pub = "nubip/ce123/panel1/data";

WiFiClient espClient;
```

```
PubSubClient client(espClient);
```

У наведеному лістингу визначено основні змінні та об’єкти, необхідні для роботи вузла. Датчик температури DS18B20 підключений до GPIO 4, а для зчитування напруги, струму та освітленості використовуються аналогові входи GPIO 34, GPIO 35 та GPIO 32. Окремо задаються параметри підключення до мережі Wi-Fi та адреса MQTT-брокера, через який вузол передаватиме телеметричні дані.

Також у цьому фрагменті задається ідентифікатор панелі та MQTT-топік, що дозволяє розрізняти повідомлення від різних вузлів у багатовузловій системі. Крім того, саме тут створюються об’єкти для роботи з датчиком температури, мережевим клієнтом і MQTT-клієнтом. Таким чином, цей фрагмент формує початкову програмну основу для подальшої реалізації всіх функцій вузла моніторингу.

Після задання основних параметрів вузла виконується початкова ініціалізація всіх необхідних підсистем. На цьому етапі запускається послідовний інтерфейс, активується датчик температури, встановлюється з’єднання з бездротовою мережею та задаються параметри підключення до MQTT-брокера. Саме ця частина програми готує вузол до подальшої роботи в циклічному режимі.

Лістинг 3.2 - Ініціалізація датчиків, Wi-Fi та MQTT

```
void setup() {
  Serial.begin(115200);
  sensors.begin();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay(500);
  client.setServer(mqtt_server, 1883);
}
```

У цьому фрагменті програма послідовно виконує всі базові дії, без яких вузол не зміг би почати роботу. Спочатку відкривається Serial Monitor, що дозволяє виводити службову інформацію під час тестування. Далі ініціалізується датчик DS18B20, після чого виконується підключення мікроконтролера до мережі Wi-Fi.

Після успішного підключення задається адреса MQTT-брокера, через який у подальшому передаватимуться телеметричні дані. Отже, цей етап забезпечує підготовку вузла до зчитування параметрів та надсилання сформованих повідомлень у мережу.

Основна частина роботи вузла пов’язана зі зчитуванням параметрів, які характеризують поточний стан сонячної панелі. На кожному циклі програма отримує температуру з цифрового датчика, а також зчитує аналогові значення, що відповідають освітленості, напрузі та струму. Саме ці дані надалі використовуються для розрахунку потужності та формування телеметричного повідомлення.

Лістинг 3.3 - Зчитування температури, освітленості, напруги та струму

```
sensors.requestTemperatures();
float temperature = sensors.getTempCByIndex(0);

int lightRaw = analogRead(lightPin);
int illumination = map(lightRaw, 4095, 0, 0, 100);
if (illumination < 0) illumination = 0;
if (illumination > 100) illumination = 100;

float maxVoltage = (analogRead(voltagePin) / 4095.0) * 24.0;
float maxCurrent = (analogRead(currentPin) / 4095.0) * 10.0;
```

У цьому фрагменті спочатку виконується оновлення вимірювання температури та отримання її значення з датчика DS18B20. Після цього програма зчитує аналогове значення освітленості та перетворює його у відсоткову шкалу від 0 до 100. Такий підхід дозволяє отримати більш наочне представлення рівня освітлення, яке потім можна використовувати в подальших розрахунках.

Окремо визначаються умовні значення максимальної напруги та струму, які зчитуються з аналогових входів. У межах моделі Wokwi це дозволяє імітувати роботу панелі за різних умов. Таким чином, у цій частині програми формується набір початкових параметрів, необхідних для обчислення фактичних електричних характеристик панелі.

Після отримання вхідних значень програма переходить до етапу обробки даних. На цьому етапі обчислюються фактичні значення напруги та струму з урахуванням освітленості, визначається потужність панелі, а потім формується структуроване повідомлення у форматі JSON. Саме цей формат використовується для подальшого передавання телеметрії до контролера-шлюзу.

Лістинг 3.4 - Обчислення потужності та формування JSON

```
float actualVoltage = maxVoltage * (illumination / 100.0);
float actualCurrent = maxCurrent * (illumination / 100.0);
float actualPower = actualVoltage * actualCurrent;


String payload = "{\"panel_id\": " + String(PANEL_ID) +
                 ", \"temp\": " + String(temperature) +
                 ", \"vol\": " + String(actualVoltage) +
                 ", \"cur\": " + String(actualCurrent) +
                 ", \"pow\": " + String(actualPower) +
                 ", \"light\": " + String(illumination) + "}";
```

У наведеному фрагменті програма спочатку визначає фактичну напругу та струм, коригуючи їх відповідно до рівня освітленості. Після цього обчислюється потужність як добуток напруги та струму. Такий підхід дозволяє пов’язати електричні параметри панелі з умовами її роботи в межах моделі.

Далі всі отримані значення об’єднуються в одне JSON-повідомлення. До нього входять ідентифікатор панелі, температура, напруга, струм, потужність та рівень освітленості. Завдяки цьому контролер-шлюз отримує вже готовий набір параметрів, який можна одразу обробляти без додаткового перетворення формату.

Після формування JSON-повідомлення виконується його передавання через MQTT. Перед відправленням програма перевіряє, чи зберігається підключення до брокера. Якщо з’єднання відсутнє, воно встановлюється повторно, після чого

повідомлення передається у відповідний топік. Одночасно ці дані виводяться в Serial Monitor, що спрощує перевірку роботи вузла під час моделювання.

Лістинг 3.5 - Надсилання MQTT-повідомлення

```
if (!client.connected()) {
  String clientId = "ESP32Panel_" + String(PANEL_ID) + "_" +
String(random(0xffff), HEX);
  client.connect(clientId.c_str());
}
client.loop();


Serial.println(payload);
client.publish(topic_pub, payload.c_str());


delay(15000);
```

У цьому фрагменті спочатку виконується перевірка MQTT-з'єднання. Якщо вузол не підключений до брокера, програма створює унікальний ідентифікатор клієнта та повторно ініціалізує з'єднання. Після цього запускається обробка MQTT-клієнта, виводиться сформоване повідомлення у Serial Monitor і виконується його публікація у визначений топік.

Завершується цикл затримкою на 15 секунд, після чого вся послідовність дій повторюється знову. Таким чином, вузол панелі регулярно передає актуальні телеметричні дані в мережу та підтримує стабільний зв'язок із контролером-шлюзом. Повний текст програми вузла моніторингу сонячної панелі наведено в додатку А.

3.2 Реалізація програмного забезпечення контролера-шлюзу

Контролер-шлюз приймає повідомлення від вузлів панелей, зберігає їх поточний стан, виконує локальний аналіз даних та надсилає результати на сервер через HTTP POST.

Для збереження стану кожної панелі використовується структура `SolarPanel`. Щоб контролер-шлюз міг працювати одночасно з кількома панелями, у програмі потрібно передбачити структуру збереження їх поточного стану. У цій структурі містяться основні параметри кожного вузла, які надходять після приймання MQTT-

повідомлень. Такий підхід дозволяє шлюзу не просто обробляти окремі пакети даних, а підтримувати актуальну інформацію про всі активні панелі системи.

Лістинг 3.6 - Структура даних панелі у шлюзі

```
struct SolarPanel {
  int id;
  float temp;
  float voltage;
  float current;
  float power;
  int light;
  int lastDbResponse;
  bool active = false;
};

SolarPanel fleet[6];
```

У цьому фрагменті задається структура `SolarPanel`, яка містить ідентифікатор панелі, температуру, напругу, струм, потужність, рівень освітленості, код останньої відповіді сервера та ознаку активності вузла. Після цього створюється масив `fleet`, у якому шлюз зберігає дані для кількох панелей одночасно.

Такий спосіб організації даних є зручним, оскільки дозволяє централізовано оновлювати параметри кожного вузла після приймання нового повідомлення. Крім того, на основі цих значень шлюз надалі може виконувати аналіз стану системи та передавати інформацію на сервер.

Після приймання та збереження параметрів панелей контролер-шлюз повинен передати цю інформацію до серверної частини. Для цього в програмі реалізовано окрему функцію, яка формує JSON-документ, додає службову інформацію про стан панелі та надсилає його через HTTP POST. Такий підхід дозволяє винести роботу з сервером в окремий програмний блок і не змішувати її з логікою приймання MQTT-повідомлень.

Лістинг 3.7 - Надсилання даних на сервер

```
void sendToDatabase(int id) {
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient wifiClient;
    HTTPClient HTTP;
    HTTP.begin(wifiClient, database_url);
    HTTP.addHeader("Content-Type", "application/json");

    StaticJsonDocument<512> dbDoc;
    dbDoc["panel_id"] = fleet[id].id;
```

```
    dbDoc["temp"] = fleet[id].temp;
    dbDoc["vol"] = fleet[id].voltage;
    dbDoc["cur"] = fleet[id].current;
    dbDoc["power"] = fleet[id].power;
    dbDoc["light"] = fleet[id].light;

    if (fleet[id].temp > 100.0) dbDoc["status"] = "WARNING: OVERHEAT";
    else dbDoc["status"] = "NORMAL";

    String body;
    serializeJson(dbDoc, body);
    fleet[id].lastDbResponse = HTTP.POST(body);
    HTTP.end();
  }
}
```

У цьому фрагменті спочатку перевіряється наявність підключення до бездротової мережі. Якщо з'єднання активне, створюється HTTP-клієнт і задається адреса серверного ресурсу, на який буде надсилатися інформація. Далі формується JSON-документ, до якого записуються основні параметри панелі: її ідентифікатор, температура, напруга, струм, потужність і рівень освітленості.

Окремо в повідомлення додається службове поле `status`, яке залежить від температурного стану панелі. Якщо температура перевищує заданий поріг, формується попередження про перегрів. В іншому випадку задається нормальний стан. Після цього JSON серіалізується в рядок, надсилається на сервер за допомогою HTTP POST, а код відповіді зберігається у структурі даних панелі. Така реалізація дозволяє не тільки передати інформацію на сервер, а й зафіксувати результат цієї операції для подальшого контролю.

Перед обробкою телеметричних даних у шлюзі додатково виконується перевірка службового ключа вузла. Для цього вузол панелі передає у JSON-повідомленні поле key, а контролер-шлюз порівнює його зі значенням, заданим у константі DEVICE_KEY. Якщо ключ відсутній або не збігається з очікуваним значенням, повідомлення відхиляється і не використовується для оновлення стану системи. Це дозволяє реалізувати базовий захист від випадкових або сторонніх MQTT-повідомлень у межах розробленої моделі.

Лістинг 3.8 - Перевірка службового ключа вузла на контролері-шлюзі

```
      const char* receivedKey = doc["key"];
      if (receivedKey == nullptr || String(receivedKey) != DEVICE_KEY) {
        Serial.println("[ЗАХИСТ] Повідомлення відхилено: неправильний ключ вузла.");
```

```
        return;
      }
```

У цьому фрагменті з отриманого JSON-повідомлення зчитується значення поля `key`. Якщо такого поля немає або воно не відповідає очікуваному службовому ключу, функція завершує обробку повідомлення за допомогою оператора `return`. У такому випадку дані не потрапляють до структури стану панелей і не передаються далі на сервер.

Однією з ключових частин програмного забезпечення шлюзу є обробка повідомлень, які надходять від вузлів панелей через MQTT. Для цього використовується окрема функція зворотного виклику, яка запускається щоразу після отримання нового повідомлення. Саме в цьому місці програма витягує параметри з JSON, визначає, від якої панелі надійшли дані, оновлює її поточний стан і за потреби ініціює передавання інформації на сервер.

Лістинг 3.9 - Приймання та обробка повідомлень від панелей

```
void callback(char* topic, byte* payload, unsigned int length) {
  StaticJsonDocument<512> doc;
  deserializeJson(doc, payload, length);
  int id = doc["panel_id"];

  if (id > 0 && id <= 5) {
    fleet[id].id = id;
    fleet[id].temp = doc["temp"];
    fleet[id].voltage = doc["vol"];
    fleet[id].current = doc["cur"];
    fleet[id].light = doc["light"];
    fleet[id].power = fleet[id].voltage * fleet[id].current;
    fleet[id].active = true;

    sendToDatabase(id);
  }
}
```

У цьому фрагменті програма спочатку виконує десеріалізацію отриманого JSON-повідомлення, після чого зчитує ідентифікатор панелі. Якщо номер панелі знаходиться в допустимому діапазоні, шлюз оновлює відповідні поля у внутрішній структурі даних: температуру, напругу, струм, рівень освітленості та потужність. Також для цієї панелі встановлюється ознака активності, що дозволяє враховувати її під час подальшого аналізу.

Після оновлення поточного стану вузла викликається функція надсилання даних на сервер. Таким чином, саме ця частина програми є основною ланкою між прийманням телеметрії від панелей і подальшою передачею інформації у серверну частину системи.

Окрім приймання та передавання даних, контролер-шлюз у розробленій системі виконує базовий аналіз поточного стану генерації. Для цього в програмі передбачено окрему функцію, яка узагальнює параметри активних панелей, визначає сумарну потужність та середній рівень освітленості, а на основі цих значень формує повідомлення про поточний режим роботи системи.

Лістинг 3.10 - Аналіз сумарної потужності та освітленості

```
void analyzeAndControl() {
  float totalPower = 0;
  int activeCount = 0;
  int totalLight = 0;

  for (int i = 1; i <= 5; i++) {
    if (fleet[i].active) {
      totalPower += fleet[i].power;
      totalLight += fleet[i].light;
      activeCount++;
    }
  }

  if (activeCount == 0) return;
  int avgLight = totalLight / activeCount;

  Serial.println("\n--- SES MESH INTELLIGENCE ---");
  Serial.printf("Grid Power: %.2f W | Weather (Sunlight): %d%%\n", totalPower,
avgLight);

  if (avgLight >= 80 && totalPower > INVERTER_MAX_POWER) {
    Serial.println("![УВАГА] Сонячний пік! Перевантаження інвертора.");
  }
  else if (avgLight <= 40 && totalPower < MIN_REQUIRED_POWER) {
    Serial.println("?[УВАГА] Сильна хмарність! Падіння генерації.");
  }
  else {
    Serial.println("[СИСТЕМА] Баланс оптимальний.");
  }
}
```

У цьому фрагменті програма послідовно переглядає всі активні панелі, підсумовує їхню потужність і накопичує значення освітленості. Після цього визначається середній рівень освітлення для всіх вузлів, які беруть участь у роботі

системи. Саме ці два параметри використовуються як основа для подальшої оцінки стану генерації.

Якщо освітленість є високою, а сумарна потужність перевищує задане граничне значення, шлюз формує попередження про можливе перевантаження. Якщо ж освітленість низька, а потужність недостатня, система повідомляє про падіння генерації. У випадку, коли значення залишаються в нормальних межах, виводиться повідомлення про збалансований режим роботи. Отже, ця частина програми показує, що шлюз у розробленій системі виконує не тільки функцію передавання даних, а й елементи локального інтелектуального аналізу.

Для того щоб під час моделювання можна було швидко оцінити стан усіх активних панелей, у програмі шлюзу передбачено виведення інформації у табличному вигляді. Такий спосіб представлення є зручним, оскільки дозволяє одночасно бачити основні параметри кожного вузла, код відповіді сервера та ознаку активності. Після формування таблиці програма також викликає функцію аналізу стану генерації.

Лістинг 3.11 - Виведення таблиці стану панелей

```
void displayTable() {
  Serial.println("\n| ID | Temp  | Volt  | Curr  | Power  | Light | DB Resp |
Status |");
  Serial.println("|----|-------|-------|-------|--------|-------|---------|----
----|");
  for (int i = 1; i <= 5; i++) {
    if (fleet[i].active) {
      Serial.printf("| %-2d | %-5.1f | %-5.1f | %-5.1f | %-6.1f | %-4d%% | %-7d
| ACTIVE |\n",
                    i, fleet[i].temp, fleet[i].voltage, fleet[i].current,
fleet[i].power, fleet[i].light, fleet[i].lastDbResponse);
    }
  }
  analyzeAndControl();
}
```

Тут спочатку виводиться заголовок таблиці, після чого програма послідовно переглядає всі панелі, дані про які зберігаються у внутрішній структурі шлюзу. Для кожного активного вузла в таблиці відображаються ідентифікатор, температура, напруга, струм, потужність, рівень освітленості, код останньої відповіді сервера та статус активності. Завдяки цьому оператор або розробник може одразу побачити поточний стан системи без додаткового аналізу JSON-повідомлень.

Після виведення таблиці викликається функція аналізу, яка формує текстовий висновок щодо режиму роботи системи. Таким чином, цей фрагмент об’єднує наочне представлення телеметричних даних і базову оцінку стану генерації, що є корисним під час тестування моделі в середовищі Wokwi.

Завершальним елементом програмної логіки шлюзу є організація його підключення до MQTT-брокера, налаштування функції обробки повідомлень та забезпечення підписки на топіки панелей. Для цього в програмі використовуються функції `setup()` і `loop()`, у яких реалізується початкове підключення до мережі, налаштування MQTT-клієнта та підтримання стабільної роботи шлюзу в циклічному режимі.

Лістинг 3.12 - Підключення шлюзу до брокера та підписка на панелі

```
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay(500);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    String clientId = "MasterGateway_" + String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) client.subscribe(topic_sub);
  }
  client.loop();

  if (millis() - lastUpdate > 5000) {
    displayTable();
    lastUpdate = millis();
  }
}
```

У цьому фрагменті спочатку виконується початкова ініціалізація шлюзу. Програма запускає послідовний інтерфейс, підключається до бездротової мережі Wi-Fi, задає адресу MQTT-брокера та прив’язує функцію `callback()`, яка надалі обробляє всі вхідні повідомлення від панелей. Після цього шлюз переходить до циклічного режиму роботи.

У функції `loop()` контролер перевіряє, чи зберігається підключення до брокера. Якщо з’єднання відсутнє, воно відновлюється, після чого виконується підписка на

груповий MQTT-топік, через який надходять дані від панелей. Далі запускається обробка MQTT-клієнта, а через заданий інтервал часу формується таблиця стану системи. Таким чином, цей фрагмент забезпечує безперервну роботу шлюзу, приймання телеметрії та регулярне оновлення інформації про стан панелей.

Повний текст програми контролера-шлюзу наведено в додатку Б. У додатку подано повну реалізацію приймання MQTT-повідомлень, обробки JSON-даних, аналізу стану системи та передавання інформації на сервер.

3.3 Реалізація формування та обробки JSON-повідомлень

У розробленій системі для передавання телеметричних даних між вузлами панелей, контролером-шлюзом та серверною частиною використовується формат JSON. Такий підхід був обраний через те, що JSON має просту структуру, легко формується у програмі мікроконтролера та зручно обробляється на стороні приймача. Для системи моніторингу це важливо, оскільки кожне повідомлення повинно містити не одне значення, а цілий набір параметрів, пов’язаних з конкретною сонячною панеллю.

На стороні вузла панелі JSON-повідомлення формується після зчитування всіх необхідних параметрів. Спочатку програма отримує температуру з датчика DS18B20, потім зчитує аналогові значення, які відповідають напрузі, струму та рівню освітленості. Після цього виконується перерахунок отриманих значень у зручний для подальшої обробки вигляд. На основі напруги та струму додатково обчислюється потужність панелі.

До складу JSON-повідомлення входять такі основні поля:

- ідентифікатор панелі;
- температура;
- напруга;
- струм;
- потужність;
- рівень освітленості.

Ідентифікатор панелі є обов’язковим елементом повідомлення, оскільки система розрахована на роботу з кількома вузлами моніторингу. Якщо передавати лише

значення параметрів без ідентифікатора, контролер-шлюз не зможе визначити, від якої саме панелі надійшли дані. Завдяки використанню окремого ідентифікатора шлюз може зберігати стан кожної панелі окремо та оновлювати тільки ті дані, які відповідають конкретному вузлу.

Приклад логічної структури повідомлення має такий вигляд:

```
{
  "panel_id": "panel_1",
  "temperature": 32.5,
  "voltage": 18.7,
  "current": 1.25,
  "power": 23.37,
  "light": 76
}
```

У наведеному прикладі показано, що одне повідомлення містить повний набір даних про поточний стан окремої сонячної панелі. Така структура є зручною для передавання через MQTT, оскільки всі параметри передаються одним пакетом і не потребують окремих повідомлень для кожного значення.

На стороні контролера-шлюзу JSON-повідомлення приймається з MQTT-топіка, після чого виконується його розбір. Програма шлюзу виділяє з повідомлення ідентифікатор панелі та основні числові параметри. Після цього отримані значення записуються у внутрішню структуру даних, яка відповідає конкретному вузлу. Такий підхід дозволяє шлюзу підтримувати актуальний стан усіх панелей, які беруть участь у роботі системи.

Окремо слід зазначити, що використання JSON спрощує подальшу передачу даних на сервер. Оскільки серверна частина також може приймати структуровані JSON-запити, шлюзу не потрібно повністю змінювати формат даних перед відправленням. Це зменшує складність програмної реалізації та робить обмін даними більш зрозумілим.

У цій системі JSON фактично використовується як спільний формат для передачі телеметрії між вузлом, шлюзом і серверною частиною. Він забезпечує зручне групування параметрів, підтримує роботу з кількома панелями та дозволяє без значних

змін передавати інформацію від вузла панелі до шлюзу, а потім від шлюзу до серверної частини.

3.4 Забезпечення стабільності обміну даними

Оскільки розроблена система використовує бездротовий обмін даними, у програмній реалізації важливо враховувати можливі переривання зв’язку. У реальних умовах Wi-Fi-з’єднання може тимчасово зникати через нестабільність мережі, перезавантаження точки доступу, зміну рівня сигналу або інші зовнішні фактори. Крім цього, MQTT-брокер або серверна частина також можуть бути тимчасово недоступними. Через це програмне забезпечення не повинно завершувати роботу після першої помилки з’єднання.

У вузлі сонячної панелі передбачено підключення до бездротової мережі Wi-Fi перед початком основного циклу роботи. Після встановлення з’єднання вузол переходить до зчитування параметрів і публікації MQTT-повідомлень. Якщо з’єднання з MQTT-брокером відсутнє, програма повинна повторно виконати підключення. Це дозволяє вузлу не втрачати працездатність після короткочасного збою.

Контролер-шлюз також потребує перевірки стану з’єднання. Він повинен бути підключений до Wi-Fi, MQTT-брокера та мати можливість надсилати HTTP-запити на сервер. Якщо MQTT-з’єднання було втрачено, шлюз повторно підключається до брокера та відновлює підписку на потрібні топіки. Завдяки цьому після відновлення зв’язку він знову може приймати повідомлення від панельних вузлів.

Для передавання даних на сервер використовується HTTP POST. Після відправлення запиту шлюз отримує код відповіді сервера. Цей код дозволяє оцінити, чи було передавання успішним. Якщо сервер повертає успішну відповідь, дані можна вважати прийнятими. Якщо ж код відповіді вказує на помилку або сервер не відповідає, у Serial Monitor виводиться відповідне повідомлення. Це спрощує налагодження системи під час моделювання.

У програмі шлюзу також передбачено зберігання поточного стану кожної панелі. Це потрібно для того, щоб система не обмежувалася обробкою лише одного останнього повідомлення. Після приймання даних від вузла шлюз оновлює відповідний запис у внутрішній структурі, де зберігаються поточні значення температури, напруги, струму,

потужності, освітленості та стан активності панелі. Завдяки цьому можна виводити загальну таблицю стану системи та виконувати базовий аналіз генерації.

Ще одним елементом стабільності є періодичність передавання телеметрії. Вузол панелі не надсилає дані безперервно, а робить це через певний інтервал часу. Такий підхід зменшує навантаження на мережу та MQTT-брокер. Для системи моніторингу сонячних панелей цього достатньо, оскільки параметри панелей не змінюються миттєво з дуже великою швидкістю. Регулярне оновлення з інтервалом у кілька секунд дозволяє бачити зміну параметрів і водночас не перевантажувати систему зайвими повідомленнями.

У межах моделі, реалізованої в середовищі Wokwi, така логіка дозволяє перевірити основні ситуації: запуск вузлів, встановлення з'єднання, передавання повідомлень, приймання даних шлюзом, відправлення інформації на сервер та виведення результатів у Serial Monitor. Це робить програмну реалізацію більш надійною та наближеною до практичного використання.

Таким чином, стабільність обміну даними в системі забезпечується за рахунок повторного підключення до мережі та MQTT-брокера, контролю відповіді сервера, збереження стану кожної панелі та використання періодичного передавання телеметрії. У сукупності ці рішення дозволяють системі продовжувати роботу навіть у разі короткочасних збоїв зв'язку.

3.5 Висновки до розділу 3

У третьому розділі було описано програмну частину системи моніторингу панелей у СЕС. Окремо розглянуто код вузла сонячної панелі та код контролера-шлюзу, оскільки в системі вони виконують різні задачі.

Для вузла сонячної панелі реалізовано підключення до Wi-Fi, зчитування температури, освітленості, напруги та струму. Після цього програма обчислює потужність панелі, формує JSON-повідомлення і передає його через MQTT. За рахунок цього кожен вузол може працювати як окреме джерело даних про стан своєї панелі.

Для контролера-шлюзу реалізовано приймання MQTT-повідомлень від панельних вузлів, розбір JSON-даних, збереження поточного стану кожної панелі та передавання інформації на сервер через HTTP POST. Також шлюз виконує просту

обробку отриманих значень: визначає сумарну потужність, середній рівень освітленості та виводить повідомлення про поточний режим роботи системи.

У розділі також описано структуру JSON-повідомлень, які використовуються для передавання телеметрії. Такий формат зручний тим, що всі основні параметри панелі передаються одним повідомленням. Це спрощує обробку даних на стороні шлюзу і не потребує окремого повідомлення для кожного параметра.

Також у програмній частині враховано ситуації, коли зв’язок може тимчасово перериватися. Для цього передбачено повторне підключення до Wi-Fi та MQTT-брокера, перевірку відповіді сервера після HTTP-запиту, періодичне передавання телеметрії та збереження актуального стану панелей на шлюзі.

У результаті програмна частина охоплює весь основний шлях проходження даних: від зчитування параметрів сонячної панелі до передавання інформації на сервер. Це дозволяє перейти до моделювання системи в Wokwi, перевірки обміну даними та аналізу отриманих результатів.

РОЗДІЛ 4 МОДЕЛЮВАННЯ, НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ СИСТЕМИ

4.1 Опис середовища моделювання Wokwi

Для перевірки працездатності розробленої системи було використано онлайн-середовище моделювання **Wokwi**. Воно дозволяє створювати віртуальні електронні схеми на базі мікроконтролерів, підключати датчики, допоміжні модулі, елементи керування та перевіряти логіку роботи програмного забезпечення без використання фізичного обладнання.

Вибір саме цього середовища зумовлений кількома причинами. По-перше, Wokwi підтримує мікроконтролери ESP32, які застосовуються в розробленій системі як для вузлів панелей, так і для контролера-шлюзу. По-друге, середовище дає змогу моделювати підключення цифрових і аналогових датчиків, а також використовувати вбудований Serial Monitor для спостереження за результатами роботи програми. По-третє, Wokwi є зручним засобом для поетапного налагодження системи, оскільки дозволяє оперативно змінювати параметри моделі та спостерігати, як на це реагує програмна логіка.

У межах дипломної роботи середовище Wokwi використано для побудови схеми вузла панелі, створення окремого вузла шлюзу, підключення датчиків і допоміжних елементів, а також для перевірки передачі телеметричних даних між елементами системи. Застосування такого підходу дало можливість реалізувати повноцінне моделювання бездротової системи моніторингу та оцінити її функціонування в умовах, максимально наближених до реальної роботи.

Отже, Wokwi у даному проєкті виступає не лише як допоміжний інструмент для перевірки коду, а як основне середовище експериментальної реалізації й тестування системи.

4.2 Побудова схеми системи у середовищі Wokwi

На етапі моделювання було створено схему вузла моніторингу сонячної панелі, яка включає мікроконтролер ESP32, датчик температури DS18B20, два потенціометри для моделювання напруги та струму, а також фоторезисторний модуль для визначення освітленості.

Основним елементом вузла є плата **ESP32**, яка виконує функції обробки даних, підключення до Wi-Fi та передачі повідомлень через MQTT. До неї підключено датчик температури DS18B20, який використовує цифрову шину OneWire. Лінія даних датчика з'єднана з виводом **GPIO 4**, а для коректної роботи шини застосовано підтягувальний резистор номіналом **4,7 кОм**.

Для моделювання електричних параметрів панелі використано два потенціометри. Один із них формує аналогове значення напруги й підключений до **GPIO 34**, другий відповідає за моделювання струму та підключений до **GPIO 35**. Такий підхід дозволяє вручну змінювати вхідні значення та спостерігати, як програма перераховує фактичні параметри панелі.

Рівень освітленості у системі задається за допомогою фоторезисторного сенсора, аналоговий вихід якого підключений до **GPIO 32**. У результаті користувач може змінювати освітлення в моделі та перевіряти, як це впливає на обчислені значення напруги, струму та потужності.

Усі елементи схеми живляться від виходу **3.3 В** мікроконтролера ESP32, а також мають спільну шину заземлення. Така конфігурація відповідає типовій структурі мікроконтролерного вузла збору даних.

Окремо в середовищі моделювався контролер-шлюз, який працює на базі іншого ESP32. Саме він приймає MQTT-повідомлення, аналізує дані та надсилає результати на сервер. У сукупності ці елементи формують завершену модель системи моніторингу панелей у СЕС.

4.3 Налаштування вузлів панелей

Налаштування вузлів панелей передбачало конфігурацію програмної та апаратної частини моделі таким чином, щоб забезпечити стабільне зчитування параметрів і коректну передачу даних у мережу.

На програмному рівні для кожного вузла задавалися параметри підключення до бездротової мережі, адреса MQTT-брокера, ідентифікатор панелі та топік для публікації телеметричних даних. Вузли панелей мають спільну логіку роботи, а їх розрізнення відбувається за рахунок змінної `PANEL_ID` та відповідного імені MQTT-каналу. Це

дозволяє використовувати один і той самий програмний шаблон для декількох панелей, змінюючи лише параметри ідентифікації.

Після запуску кожен вузол автоматично виконує підключення до мережі Wi-Fi, ініціалізацію датчика температури та налаштування MQTT-клієнта. Далі система переходить у циклічний режим, у якому панель періодично зчитує поточні значення датчиків і формує JSON-повідомлення.

На етапі тестування в середовищі Wokwi було використано від однієї до двох панелей, хоча загальна логіка шлюзу передбачає підтримку до п’яти вузлів. Такий підхід дозволив перевірити як мінімальний варіант роботи системи, так і її поведінку за наявності декількох джерел телеметрії.

Таким чином, налаштування вузлів панелей підтвердило можливість масштабування системи без суттєвих змін у програмній архітектурі.

Схему моделювання системи у середовищі Wokwi наведено в додатку В.

4.4 Налаштування контролера-шлюзу

Налаштування контролера-шлюзу виконувалося окремо від вузлів панелей. Для його роботи також використовувався ESP32, на якому було реалізовано програму приймання, обробки та пересилання даних.

На початковому етапі у програмі шлюзу задавалися параметри підключення до тієї самої бездротової мережі, що і у вузлів панелей. Після виходу в мережу виконувався зв’язок із MQTT-брокером та підписка на груповий топік, який охоплює всі панелі системи. Завдяки використанню узагальненого шаблону топіка шлюз міг автоматично приймати дані від усіх вузлів без окремого налаштування для кожної панелі.

Окремим параметром у конфігурації шлюзу була адреса серверної точки для надсилання HTTP POST-запитів. У межах моделювання для цього використовувався сервіс **webhook.site**, який дозволив зручно перевірити факт передачі даних, їх формат і вміст.

Для локального контролю роботи шлюзу використовувався Serial Monitor. У ньому виводилися таблиця з поточним станом панелей, код відповіді сервера, а також блок аналітичної інформації про сумарну потужність системи та освітленість. Це значно спростило тестування та наочну перевірку результатів роботи.

Отже, налаштування шлюзу забезпечило його функціонування як центрального вузла збору, аналізу та пересилання інформації у розробленій системі моніторингу.

4.5 Сценарії тестування системи

Для перевірки працездатності розробленої системи було визначено кілька основних сценаріїв тестування. Вони охоплюють запуск вузлів, підключення до бездротової мережі, формування повідомлень, передавання даних через MQTT, приймання інформації контролером-шлюзом та надсилання даних на сервер через HTTP.

Такий підхід потрібний для того, щоб перевірити не тільки окремі елементи системи, а й увесь шлях проходження даних. У цій роботі важливо було переконатися, що значення не просто зчитуються з датчиків, а доходять до шлюзу, правильно обробляються і після цього передаються на серверну частину.

Таблиця 4.1 - Сценарії тестування системи моніторингу

№	Сценарій тестування	Що перевіряється	Очікуваний результат
1	Запуск вузла сонячної панелі	Ініціалізація ESP32, датчиків і підключення до Wi-Fi	Вузол запускається і підключається до мережі
2	Зчитування параметрів панелі	Отримання температури, напруги, струму та освітленості	У програмі формуються актуальні значення параметрів
3	Формування JSON-повідомлення	Правильність структури повідомлення з телеметрією	Повідомлення містить ідентифікатор панелі та всі основні параметри
4	Передавання даних через MQTT	Публікація повідомлення у відповідний MQTT-топік	Контролер-шлюз отримує повідомлення від вузла
5	Обробка даних на шлюзі	Розбір JSON і оновлення стану панелі	Дані правильно записуються у внутрішню структуру шлюзу
6	Передавання даних на сервер	Надсилання HTTP POST-запиту	Сервер приймає JSON-дані від шлюзу
7	Аналіз режиму роботи системи	Обчислення сумарної потужності та середньої освітленості	Шлюз формує повідомлення про нормальний режим або попередження
8	Виведення результатів у Serial Monitor	Відображення поточного стану панелей	У консолі видно таблицю параметрів і службові повідомлення

Перший сценарій стосується запуску вузла сонячної панелі. На цьому етапі перевіряється, чи коректно ініціалізується мікроконтролер ESP32, чи запускаються потрібні бібліотеки та чи встановлюється підключення до Wi-Fi. Якщо вузол не підключається до мережі, подальша передача даних через MQTT буде неможливою.

Другий сценарій пов’язаний зі зчитуванням параметрів панелі. У моделі перевіряється отримання температури з датчика DS18B20, а також зчитування аналогових значень, які відповідають напрузі, струму та освітленості. Це дозволяє переконатися, що вузол отримує всі потрібні дані перед формуванням повідомлення.

Третій сценарій перевіряє формування JSON-повідомлення. У ньому мають бути присутні ідентифікатор панелі, температура, напруга, струм, потужність та рівень освітленості. Саме наявність ідентифікатора дозволяє шлюзу розрізняти дані від різних вузлів.

Четвертий сценарій стосується передавання повідомлення через MQTT. Вузол панелі публікує дані у відповідний топік, а контролер-шлюз повинен отримати це повідомлення. Якщо повідомлення не доходить до шлюзу, це може свідчити про помилку в підключенні до MQTT-брокера, неправильний топік або проблему з мережею.

П’ятий сценарій перевіряє роботу контролера-шлюзу після отримання повідомлення. Шлюз повинен розібрати JSON-дані, визначити ідентифікатор панелі та оновити відповідні значення у внутрішній структурі. Завдяки цьому система може зберігати актуальний стан кожного вузла окремо.

Шостий сценарій пов’язаний з передаванням даних на сервер через HTTP POST. На цьому етапі перевіряється, чи формує шлюз запит, чи надсилає його на серверну адресу та чи отримує код відповіді. Це дозволяє оцінити, чи проходить передача даних за межі локальної логіки системи.

Сьомий сценарій перевіряє базовий аналіз стану системи. Контролер-шлюз обчислює сумарну потужність активних панелей і середній рівень освітленості. На основі цих значень він виводить повідомлення про нормальний режим роботи, можливе перевантаження або недостатню генерацію.

Восьмий сценарій стосується виведення результатів у Serial Monitor. Це потрібно для зручної перевірки роботи системи під час моделювання. У консолі мають відображатися прийняті параметри, службові повідомлення, коди відповідей сервера та підсумкова інформація про стан системи.

Проведення таких сценаріїв дозволяє перевірити роботу системи послідовно: від запуску окремого вузла до отримання даних серверною частиною. Це також спрощує пошук помилок, оскільки кожен етап можна перевіряти окремо.

4.6 Перевірка передавання даних у бездротовій мережі

Одним із головних етапів тестування була перевірка самого факту передавання даних від вузла панелі до контролера-шлюзу. Для цього після запуску програми панелі спостерігалися повідомлення, що виводяться у Serial Monitor.

У ході перевірки було встановлено, що вузол панелі після підключення до мережі Wi-Fi та MQTT-брокера починає регулярно формувати JSON-повідомлення з поточними параметрами. У виведенні серійного монітора відображалися поля `PANEL_ID`, `temp`, `vol`, `cur`, `pow` та `light`. Це підтвердило, що програма коректно зчитує значення з датчиків і виконує формування телеметричного пакета.

Після цього перевірялася реакція шлюзу на вхідні повідомлення. У Serial Monitor шлюзу було видно, що дані від панелі приймаються, розбираються та відображаються у відповідних стовпцях таблиці. Таким чином, було підтверджено працездатність каналу передавання даних у межах бездротової частини системи.

Також тестування показало, що повторне передавання даних здійснюється циклічно, а значення у таблиці шлюзу оновлюються відповідно до змін параметрів панелі. Це свідчить про стабільність періодичного обміну телеметрією між вузлами системи.

Приклади повідомлень, отриманих у процесі моделювання, наведено в додатку Г.

4.7 Перевірка обміну даними між панелями та шлюзом

Наступним етапом стала перевірка правильності обробки даних на рівні шлюзу. Для цього аналізувалося, як саме прийняті повідомлення відображаються у внутрішній логіці системи та в табличному представленні.

У процесі тестування було встановлено, що шлюз коректно визначає ідентифікатор панелі, з якої надійшло повідомлення, після чого оновлює дані відповідного вузла у внутрішньому масиві структур. У таблиці відображалися

температура, напруга, струм, потужність, освітленість, код відповіді сервера та статус активності панелі.

За умови використання лише однієї панелі таблиця містила один рядок. Під час додавання інших вузлів таблиця розширювалася по вертикалі, що підтверджує підтримку багатовузлового режиму роботи. Таким чином, було перевірено можливість системи одночасно обробляти дані від кількох джерел та зберігати їх у структурованому вигляді.

Додатково було підтверджено, що шлюз здатний незалежно обчислювати потужність на основі отриманих значень напруги та струму. Це забезпечує додатковий рівень контролю коректності телеметрії, що надходить від вузлів панелей.

Отже, перевірка обміну даними між панелями та шлюзом показала, що централізований збір та обробка телеметрії працюють стабільно і відповідають запроєктованій логіці системи.

Додаткові результати виведення телеметричних даних у консольному інтерфейсі шлюзу подано в додатку Г.

4.8 Перевірка передавання даних на сервер

Після підтвердження внутрішнього обміну даними між панелями та шлюзом було виконано тестування передавання даних на сервер. Для цього використовувався сервіс **webhook.site**, який надає можливість переглядати вміст вхідних HTTP POST-запитів у реальному часі.

У ході тестування було встановлено, що після приймання повідомлення від панелі шлюз формує окремий JSON-документ і надсилає його на серверну адресу. На стороні веб-сервісу були отримані HTTP POST-запити з типом вмісту `Application/json`, що підтверджує коректне встановлення заголовків запиту.

У тілі запиту відображалися всі необхідні параметри панелі:

- ідентифікатор;
- температура;
- напруга;
- струм;

- потужність;
- рівень освітленості;
- статус роботи.

Крім числових параметрів, у JSON також передавалося поле `status`, яке формувалося на шлюзі на основі аналізу температурного стану панелі. Це підтвердило, що сервер отримує не лише первинні дані, а й додаткову інтерпретаційну інформацію.

У таблиці шлюзу також відображався код відповіді сервера, що дозволило перевірити успішність HTTP-запиту. Завдяки цьому було підтверджено повну працездатність ланцюга передавання даних від панелі до серверної частини через контролер-шлюз.

Приклади HTTP POST-запитів і вміст JSON-повідомлень, отриманих серверною частиною, наведено в додатку Д.

4.9 Аналіз результатів моделювання

Після побудови схеми у середовищі Wokwi та налаштування програмного забезпечення вузлів панелей і контролера-шлюзу було проведено перевірку роботи всієї системи в комплексі. Основна увага під час аналізу результатів моделювання приділялася тому, чи правильно система виконує послідовність дій: зчитує параметри, формує повідомлення, передає їх через MQTT, приймає на шлюзі, обробляє та надсилає на сервер.

У процесі моделювання було встановлено, що вузол панелі стабільно виконує зчитування всіх заданих параметрів. Значення температури надходять із цифрового датчика DS18B20, а значення напруги, струму та освітленості зчитуються через аналогові входи ESP32. Після цього програма виконує обчислення потужності панелі та формує структуроване повідомлення у форматі JSON. Було перевірено реакцію контролера-шлюзу на перегрів однієї з панелей. Для цього в моделі було задано підвищене значення температури панелі, після чого шлюз прийняв телеметричні дані, оновив таблицю поточного стану та визначив статус панелі як `OVERHEAT`.

```
ID | Temp | Volt | Curr | Power | Light | DB | Status
-----------------------------------------------------
 1 | 125.0 |   3.9 |   1.7 |    6.4 |    75 | 200 | OVERHEAT

--- SES MESH INTELLIGENCE ---
Grid Power: 6.45 W | Weather (Sunlight): 75%
[СИСТЕМА] Баланс оптимальний.
[УВАГА] Панель 1 перегріта! Температура = 125.00 C
-----------------------------
```

Рисунок 4.1 - Виявлення перегріву панелі у Serial Monitor контролера-шлюзу

При температурі панелі 125 °C система відображає статус OVERHEAT у таблиці та формує окреме попередження про перегрів. Це підтверджує, що контролер-шлюз виконує не лише приймання й передавання телеметрії, а й первинний аналіз стану окремого вузла моніторингу.

Також було перевірено реакцію системи на зниження освітленості, що в межах моделі відповідає умовам сильної хмарності або затінення панелей. Для цього рівень освітленості було зменшено до низького значення, після чого потужність панелей також знизилася. Контролер-шлюз визначив такий режим як падіння генерації.

```
ID | Temp | Volt | Curr | Power | Light | DB | Status
-----------------------------------------------------
 1 |  41.6 |   0.0 |   0.0 |    0.0 |     2 | 200 | NORMAL
 2 |  47.7 |   0.6 |   0.2 |    0.1 |     9 | 200 | NORMAL

--- SES MESH INTELLIGENCE ---
Grid Power: 0.13 W | Weather (Sunlight): 5%
[УВАГА] Сильна хмарність! Падіння генерації.
```

Рисунок 4.2 - Виявлення сильної хмарності та падіння генерації у Serial Monitor

При середньому рівні освітленості 5 % сумарна потужність системи становить лише 0,13 Вт. У такому режимі контролер-шлюз формує попередження про сильну хмарність і падіння генерації. Це показує, що система враховує не лише окремі значення датчиків, а й загальний стан виробітку електроенергії.

Попередження виводиться у Serial Monitor контролера-шлюзу після обробки отриманих телеметричних даних. Такий результат свідчить про те, що логіка аналізу на

рівні шлюзу працює правильно, а система може не тільки приймати числові параметри, а й формувати коротку оцінку поточного стану генерації.

Окремо було проаналізовано роботу контролера-шлюзу. Під час моделювання він коректно підключався до тієї самої бездротової мережі, що й вузли панелей, після чого виконував підписку на потрібні MQTT-топіки. Після надходження повідомлення від панелі шлюз правильно визначав її ідентифікатор, зчитував передані значення з JSON-повідомлення та оновлював внутрішню структуру даних, у якій зберігався поточний стан активних вузлів. Це підтверджує, що обраний підхід до централізованої обробки телеметрії працює коректно і дозволяє підтримувати актуальний стан системи.

Важливим результатом моделювання стало також підтвердження працездатності механізму виведення поточних параметрів у табличному вигляді. У Serial Monitor шлюзу відображалися ідентифікатор панелі, температура, напруга, струм, потужність, рівень освітленості, код відповіді сервера та статус активності. Такий спосіб представлення інформації є зручним, оскільки дозволяє швидко оцінити стан усіх підключених вузлів без додаткової обробки даних. Якщо в системі активна лише одна панель, таблиця містить один рядок, а при збільшенні кількості вузлів вона автоматично розширюється. Це показує, що логіка масштабування в межах закладеної моделі реалізована правильно.

Під час аналізу було також перевірено коректність роботи локального обчислення потужності та узагальнених параметрів на рівні шлюзу. Програма шлюзу виконує підсумовування потужності активних панелей і визначає середній рівень освітленості. На основі цих значень формується короткий текстовий висновок щодо стану системи. Якщо освітленість висока, а сумарна потужність перевищує заданий поріг, система видає попередження про можливе перевантаження. Якщо освітленість низька, а потужність недостатня, формується повідомлення про зниження генерації. У випадках, коли значення параметрів залишаються в допустимих межах, система повідомляє про нормальний режим роботи. Це свідчить про те, що шлюз виконує не лише передавання даних, а й базовий аналіз стану мережі панелей.

Ще одним важливим результатом стало підтвердження працездатності передавання даних на сервер. Після приймання телеметрії від панелі шлюз формував

окремий JSON-документ і надсилав його через HTTP POST. У сервісі webhook.site було зафіксовано успішне надходження запиту, а також відображено вміст переданого JSON. Це означає, що заключний етап роботи системи, а саме передавання обробленої інформації до зовнішнього серверного ресурсу, також реалізований правильно.

За результатами проведеного моделювання можна зробити висновок, що всі основні етапи роботи системи функціонують узгоджено. Вузол панелі виконує зчитування та формування телеметрії, контролер-шлюз приймає й обробляє ці дані, а серверна частина отримує підготовлену інформацію для подальшого зберігання або візуалізації. При цьому система зберігає логічну послідовність роботи і не потребує ручного втручання після початкового запуску.

Разом із тим результати моделювання показують, що розроблена система є саме функціональною моделлю, а не завершеним промисловим рішенням. У поточному варіанті частина параметрів імітується в середовищі Wokwi за допомогою потенціометрів і фоторезисторного модуля, а роль серверної частини виконує тестовий вебресурс. Однак для дипломної роботи це є достатнім, оскільки дозволяє перевірити правильність архітектури системи, логіку взаємодії між її елементами та працездатність програмної реалізації.

За результатами моделювання можна зробити висновок, що система працює в межах поставленої задачі: вузли передають дані, шлюз їх приймає, обробляє і надсилає на сервер.

4.10 Оцінка можливостей масштабування системи

Під час розробки бездротової системи моніторингу панелей у СЕС важливо було врахувати не лише її працездатність у поточній конфігурації, а й можливість подальшого розширення. Це пов’язано з тим, що в реальних умовах сонячна електростанція рідко обмежується однією або двома панелями. Як правило, така система складається з більшої кількості модулів, тому архітектура моніторингу повинна допускати збільшення числа вузлів без повного перепроєктування.

При збільшенні кількості панелей важливим стає не тільки додавання нових вузлів моніторингу, а й спосіб організації зв’язку між ними. У невеликій моделі

достатньо прямого передавання даних від кожного вузла до контролера-шлюзу. Але для більшої СЕС така схема може мати обмеження, оскільки окремі панелі можуть бути розташовані далі від шлюзу або працювати в умовах нестабільного сигналу. У цьому випадку систему можна розширити за рахунок mesh-мережі, де частина вузлів буде виконувати роль проміжних ретрансляторів.

Перехід до mesh-структури не змінює загальну логіку роботи системи. Вузли так само зчитують параметри панелей, формують повідомлення у форматі JSON і передають телеметрію для подальшої обробки. Відмінність полягає лише в маршруті передавання даних: повідомлення може пройти через один або кілька проміжних вузлів, перш ніж потрапити до контролера-шлюзу. Завдяки цьому система стає більш придатною для розгортання на більших об’єктах, де прямий зв’язок кожного вузла зі шлюзом не завжди є зручним або стабільним.

У поточній реалізації для перевірки працездатності залишено саме базову централізовану схему, оскільки вона достатня для демонстрації збору, передавання та обробки телеметрії.

У межах розробленої моделі кожна панель представлена окремим вузлом на базі ESP32, який має власний ідентифікатор і передає дані у відповідний MQTT-топік. Такий підхід спрощує масштабування, оскільки для додавання нового вузла не потрібно змінювати загальну логіку роботи всієї системи. Достатньо створити ще один однотипний модуль, призначити йому новий ідентифікатор та відповідний канал передавання даних. Це означає, що архітектурно система вже побудована з урахуванням можливого розширення.

Важливу роль у цьому відіграє використання MQTT. Принцип “публікація - підписка” дозволяє організувати обмін даними між багатьма вузлами без прямого з’єднання кожного пристрою з кожним. У такій схемі всі панелі виступають джерелами повідомлень, а контролер-шлюз підписується на потрібну групу топіків і отримує дані централізовано. Завдяки цьому збільшення кількості панелей не вимагає суттєвої зміни способу взаємодії між елементами системи. Змінюється лише кількість джерел телеметрії, але не сам принцип роботи мережі.

У поточній реалізації у шлюзі передбачено обробку даних від кількох панелей. Це видно із структури внутрішнього масиву, який зберігає стан активних вузлів. У моделі фактично було використано дві панелі, однак програмна частина шлюзу дозволяє підтримувати до п’яти вузлів без зміни основної логіки. Це свідчить про те, що навіть у поточному варіанті система має певний резерв для розширення.

Разом із тим під час оцінки масштабування потрібно враховувати не лише логічну, а й технічну сторону. При збільшенні кількості панелей зростає обсяг телеметричних даних, які одночасно надходять до шлюзу. У простій моделі, де повідомлення передаються з певним інтервалом, це не створює критичних труднощів. Але у більшій системі потрібно буде враховувати частоту надсилання повідомлень, стабільність Wi-Fi-з’єднання, навантаження на MQTT-брокер і швидкість обробки даних на рівні шлюзу.

Ще один аспект масштабування пов’язаний із серверною частиною. У поточному варіанті шлюз надсилає дані на тестовий вебресурс, чого достатньо для перевірки працездатності. Однак у разі збільшення кількості панелей і обсягу повідомлень доцільно буде використовувати окрему базу даних або спеціалізований серверний застосунок, який зможе не лише приймати дані, а й накопичувати їх, формувати статистику, будувати графіки та зберігати історію змін параметрів для кожного вузла.

Крім кількісного розширення системи, можливе і функціональне масштабування. Наприклад, до вузлів моніторингу можна додати нові датчики, що дозволять контролювати додаткові параметри. Це можуть бути датчики вологості, навантаження, напруги акумуляторів або якості зв’язку. Оскільки в системі використовується JSON, додавання нових полів до повідомлень не потребує повного переписування архітектури, а лише незначних змін у програмному забезпеченні вузлів і шлюзу.

Перспективним напрямом масштабування є також розширення функцій самого контролера-шлюзу. У поточній моделі він виконує базовий аналіз стану системи, але надалі його можна доповнити збереженням часових міток, формуванням журналу подій, автоматичними сповіщеннями або інтеграцією з вебінтерфейсом для перегляду даних у

реальному часі. Це зробить систему не просто засобом передавання телеметрії, а більш повноцінним інструментом експлуатаційного контролю.

Тому, розроблена система має добрі можливості для подальшого масштабування. Вона допускає збільшення кількості вузлів, розширення переліку контрольованих параметрів і вдосконалення серверної частини без кардинальної зміни загальної архітектури. Це підтверджує, що обраний підхід є не лише придатним для навчального моделювання, а й перспективним з погляду подальшого розвитку.

4.11 Висновки до розділу 4

У четвертому розділі було виконано моделювання, налаштування та тестування розробленої бездротової системи моніторингу панелей у СЕС у середовищі Wokwi. Під час роботи було перевірено правильність побудови схеми, працездатність вузлів панелей, роботу контролера-шлюзу, а також передачу даних між усіма елементами системи.

У ході моделювання підтверджено, що вузли панелей коректно зчитують температуру, освітленість, напругу та струм, після чого обчислюють потужність і формують структуровані JSON-повідомлення. Також встановлено, що контролер-шлюз стабільно приймає дані від кількох вузлів, зберігає їх поточний стан, виконує базовий аналіз параметрів і надсилає підготовлену інформацію на сервер через HTTP POST.

Окремо було перевірено виведення результатів роботи системи у Serial Monitor. Показано, що дані панелей можуть бути представлені у зручному табличному вигляді, а логіка аналізу дозволяє формувати службові повідомлення про стан системи залежно від рівня освітленості та сумарної потужності генерації.

Під час тестування також підтверджено працездатність передачі даних на серверний ресурс. Сервіс webhook.site успішно приймав HTTP POST-запити від контролера-шлюзу та відображав переданий JSON-документ. Це свідчить про те, що система забезпечує повний цикл роботи: від зчитування параметрів на вузлі панелі до передавання обробленої інформації у зовнішню серверну частину.

Аналіз результатів моделювання показав, що запропонована система є функціонально завершеною моделлю бездротового моніторингу панелей у СЕС. Вона

дозволяє реалізувати збирання, передавання, приймання, обробку та подальше надсилання телеметричних даних без необхідності складної кабельної інфраструктури.

Крім того, у розділі було оцінено можливості масштабування системи. Встановлено, що обрана архітектура дозволяє додавати нові вузли моніторингу, розширювати перелік контрольованих параметрів і вдосконалювати серверну частину без кардинальної зміни основної логіки роботи. Це підтверджує, що розроблене рішення придатне не лише для навчального моделювання, а й як основа для подальшого розвитку.

ВИСНОВКИ

У дипломній роботі було розроблено бездротову систему моніторингу панелей у сонячній електростанції, яка забезпечує зчитування основних параметрів роботи окремих модулів, передавання даних до контролера-шлюзу та подальше надсилання інформації на сервер.

У ході виконання роботи було проаналізовано предметну область, пов'язану з експлуатацією сонячних електростанцій, та визначено, що для оцінки стану панелей доцільно контролювати температуру, освітленість, напругу, струм і потужність. Показано, що контроль лише загальних показників станції не завжди дозволяє своєчасно виявити локальні проблеми окремих панелей, тому більш ефективним є підхід із використанням окремих вузлів моніторингу.

На основі проведеного аналізу було обґрунтовано вибір апаратної платформи ESP32, яка поєднує достатню продуктивність, вбудовану підтримку Wi-Fi та зручність програмної реалізації. Також було обґрунтовано використання протоколу MQTT для внутрішнього обміну телеметричними даними між панелями та шлюзом, а протоколу HTTP - для подальшої передачі обробленої інформації на сервер.

У роботі спроєктовано загальну структуру системи, яка складається з вузлів моніторингу сонячних панелей, контролера-шлюзу та серверної частини. Для вузла панелі реалізовано функції зчитування температури, освітленості, напруги та струму, обчислення потужності, формування JSON-повідомлення та його надсилання через MQTT. Для контролера-шлюзу реалізовано приймання повідомлень від кількох панелей, оновлення поточного стану вузлів, базову обробку даних, виведення результатів у табличному вигляді та передачу інформації на сервер.

Окрему увагу було приділено програмній реалізації системи. У роботі наведено основні фрагменти програмного коду вузла сонячної панелі та контролера-шлюзу, які відображають логіку роботи системи. Використання бібліотек WiFi, PubSubClient, OneWire, DallasTemperature, ArduinoJson та HTTPClient дозволило реалізувати всі основні функції без істотного ускладнення структури програмного забезпечення.

Практична перевірка роботи системи була виконана в середовищі Wokwi. У процесі моделювання підтверджено, що вузли панелей коректно зчитують параметри,

формують повідомлення у форматі JSON і передають їх через MQTT. Також підтверджено, що контролер-шлюз правильно приймає ці дані, обробляє їх і надсилає на сервер через HTTP POST. Окремо перевірено відображення результатів у Serial Monitor та приймання запитів на серверному ресурсі webhook.site.

Аналіз результатів моделювання показав, що розроблена система є працездатною в межах поставленої задачі та виконує повний цикл обробки даних: від зчитування параметрів до їх передачі у зовнішню серверну частину. Також встановлено, що обрана архітектура дозволяє масштабувати систему шляхом додавання нових вузлів моніторингу та розширення переліку контрольованих параметрів без повної зміни основної логіки роботи.

Практичне значення одержаних результатів полягає у створенні функціональної моделі бездротової системи моніторингу панелей у СЕС, яка може бути використана як основа для подальшої розробки. У подальшому система може бути доповнена підключенням реальних датчиків, використанням повноцінної бази даних, створенням вебінтерфейсу для візуалізації результатів та розширенням функцій аналітичної обробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Espressif Systems. ESP32 Series Datasheet. - Espressif Systems, 2025.
URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

2. Bluetooth SIG. The Bluetooth® Low Energy Primer. - Bluetooth Special Interest Group, 2024.
URL: https://www.bluetooth.com/wp-content/uploads/2022/05/the-bluetooth-le-primer-v1.2.0.pdf

3. Connectivity Standards Alliance. Zigbee | Complete IoT Solution. - Connectivity Standards Alliance.
URL: https://csa-iot.org/all-solutions/zigbee/

4. LoRa Alliance. What is LoRaWAN® Specification. - LoRa Alliance.
URL: https://lora-alliance.org/about-lorawan-old/

5. MDN Web Docs. An overview of HTTP. - Mozilla Foundation.
URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview

6. OASIS. MQTT Version 5.0. - OASIS Open, 2019.
URL: https://www.oasis-open.org/standard/mqtt-v5-0-os/

7. Жураковський Б. Ю. Технології Інтернету речей. - Київ, 2021.
URL: https://nubip.edu.ua/sites/default/files/u34/tehnologiyi_interntu_rechey.pdf

8. Електростанції з відновлюваними джерелами: навчальна дисципліна. - НУБіП України, 2025.
URL:
https://nubip.edu.ua/sites/default/files/rich_text_files/elektrostanciyi_z_vidnovlyuvalnimi_dzherelami_0.pdf

9. Wokwi Docs. ESP32 Simulation. - Wokwi.
URL: https://docs.wokwi.com/guides/esp32

10. PVEducation. PV Module Temperature. - PVEducation.
URL: https://www.pveducation.org/pvcdrom/modules-and-arrays/pv-module-temperature

11. Analog Devices. DS18B20 Programmable Resolution 1-Wire Digital Thermometer. - Analog Devices.
URL: https://www.analog.com/en/products/ds18b20.html

12. Arduino. Arduino IDE. - Arduino.
URL: https://www.arduino.cc/en/software

ДОДАТКИ

ДОДАТОК А - КОД ВУЗЛА ПАНЕЛІ

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>

const int oneWireBus = 4;
const int voltagePin = 34;
const int currentPin = 35;
const int lightPin = 32;

OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);

const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* mqtt_server = "broker.hivemq.com";

// Налаштування вузла
const char* DEVICE_KEY = "nubip_ses_key_2026";
const int PANEL_ID = 1;
const char* topic_pub = "nubip/ce123/panel1/data";

WiFiClient espClient;
PubSubClient client(espClient);

void connectWiFi() {
  Serial.print("WiFi connecting");
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi connected");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
}
```

```
void connectMQTT() {
  while (!client.connected()) {
    String clientId = "ESP32Panel_" + String(PANEL_ID) + "_" + String(random(0xffff),
HEX);

    Serial.print("MQTT connecting... ");
    bool ok = client.connect(clientId.c_str());

    if (ok) {
      Serial.println("OK");
    } else {
      Serial.print("FAIL, rc=");
      Serial.println(client.state());
      delay(2000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  sensors.begin();
  connectWiFi();
  client.setServer(mqtt_server, 1883);
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  if (!client.connected()) {
    connectMQTT();
  }

  client.loop();

  sensors.requestTemperatures();
  float temperature = sensors.getTempCByIndex(0);

  int lightRaw = analogRead(lightPin);
  int illumination = map(lightRaw, 4095, 0, 0, 100);
```

```
  if (illumination < 0) illumination = 0;
  if (illumination > 100) illumination = 100;

  float maxVoltage = (analogRead(voltagePin) / 4095.0) * 24.0;
  float maxCurrent = (analogRead(currentPin) / 4095.0) * 10.0;

  float actualVoltage = maxVoltage * (illumination / 100.0);
  float actualCurrent = maxCurrent * (illumination / 100.0);
  float actualPower = actualVoltage * actualCurrent;

  String payload = "{\"PANEL_ID\": " + String(PANEL_ID) +
                ", \"key\": \"" + String(DEVICE_KEY) + "\"" +
                ", \"temp\": " + String(temperature) +
                ", \"vol\": " + String(actualVoltage) +
                ", \"cur\": " + String(actualCurrent) +
                ", \"pow\": " + String(actualPower) +
                ", \"light\": " + String(illumination) + "}";

  Serial.println("Payload:");
  Serial.println(payload);

  bool sent = client.publish(topic_pub, payload.c_str());

  Serial.print("Publish result: ");
  Serial.println(sent ? "OK" : "FAIL");

  delay(15000);
}
```

ДОДАТОК Б - КОД КОНТРОЛЕРА-ШЛЮЗУ

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

const char* DEVICE_KEY = "nubip_ses_key_2026";
const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 1883;

const char* topic_sub = "nubip/ce123/+/data";
const char* database_url = "http://webhook.site/b442ee10-b122-4a4a-8e0f-d79cc6f27109";
```

```
const float INVERTER_MAX_POWER = 250.0;
const float MIN_REQUIRED_POWER = 100.0;
const float OVERHEAT_TEMP = 100.0;

WiFiClient espClient;
PubSubClient client(espClient);

struct PanelData {
  int id;
  float temp;
  float voltage;
  float current;
  float power;
  int light;
  int lastDbResponse;
  bool active;
};

PanelData fleet[6];

void connectWiFi() {
  Serial.print("WiFi connecting");
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi connected");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
}

void connectMQTT() {
  while (!client.connected()) {
    String clientId = "MasterGateway_" + String(random(0xffff), HEX);

    Serial.print("MQTT connecting... ");
    bool ok = client.connect(clientId.c_str());

    if (ok) {
```

```
      Serial.println("OK");

      bool sub = client.subscribe(topic_sub);
      Serial.print("Subscribe to ");
      Serial.print(topic_sub);
      Serial.print(": ");
      Serial.println(sub ? "OK" : "FAIL");
    } else {
      Serial.print("FAIL, rc=");
      Serial.println(client.state());
      delay(2000);
    }
  }
}

void sendToDatabase(int id) {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("HTTP skipped: WiFi not connected");
    return;
  }

  WiFiClient wifiClient;
  HTTPClient http;

  http.begin(wifiClient, database_url);
  http.addHeader("Content-Type", "application/json");

  StaticJsonDocument<512> dbDoc;
  dbDoc["panel_id"] = fleet[id].id;
  dbDoc["temp"] = fleet[id].temp;
  dbDoc["vol"] = fleet[id].voltage;
  dbDoc["cur"] = fleet[id].current;
  dbDoc["power"] = fleet[id].power;
  dbDoc["light"] = fleet[id].light;

  if (fleet[id].temp > OVERHEAT_TEMP) {
    dbDoc["status"] = "WARNING: OVERHEAT";
  } else {
    dbDoc["status"] = "NORMAL";
  }

  String body;
  serializeJson(dbDoc, body);
```

```
  Serial.println("HTTP POST body:");
  Serial.println(body);

  int code = http.POST(body);
  fleet[id].lastDbResponse = code;

  Serial.print("HTTP response code: ");
  Serial.println(code);

  http.end();
}

void displayTable() {
  Serial.println();
  Serial.println("ID | Temp | Volt | Curr | Power | Light | DB | Status");
  Serial.println("-------------------------------------------------------");

  for (int i = 1; i <= 5; i++) {
    if (fleet[i].active) {
      String status = "NORMAL";
      if (fleet[i].temp > OVERHEAT_TEMP) {
        status = "OVERHEAT";
      }

      Serial.printf("%2d | %5.1f | %5.1f | %5.1f | %6.1f | %5d | %3d | %s\n",
                    fleet[i].id,
                    fleet[i].temp,
                    fleet[i].voltage,
                    fleet[i].current,
                    fleet[i].power,
                    fleet[i].light,
                    fleet[i].lastDbResponse,
                    status.c_str());
    }
  }

  Serial.println();
}

void analyzeAndControl() {
  float totalPower = 0;
  int activeCount = 0;
```

```
  int totalLight = 0;

  for (int i = 1; i <= 5; i++) {
    if (fleet[i].active) {
      totalPower += fleet[i].power;
      totalLight += fleet[i].light;
      activeCount++;
    }
  }

  if (activeCount == 0) {
    Serial.println("No active panels");
    return;
  }

  int avgLight = totalLight / activeCount;

  Serial.println("--- SES MESH INTELLIGENCE ---");
  Serial.printf("Grid Power: %.2f W | Weather (Sunlight): %d%%\n", totalPower, avgLight);

  if (avgLight >= 80 && totalPower > INVERTER_MAX_POWER) {
    Serial.println("[УВАГА] Сонячний пік! Можливе перевантаження інвертора.");
  } else if (avgLight <= 40 && totalPower < MIN_REQUIRED_POWER) {
    Serial.println("[УВАГА] Сильна хмарність! Падіння генерації.");
  } else {
    Serial.println("[СИСТЕМА] Баланс оптимальний.");
  }

  for (int i = 1; i <= 5; i++) {
    if (fleet[i].active && fleet[i].temp > OVERHEAT_TEMP) {
      Serial.printf("[УВАГА] Панель %d перегріта! Температура = %.2f C\n", fleet[i].id,
fleet[i].temp);
    }
  }

  Serial.println("-----------------------------");
  Serial.println();
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message received on topic: ");
  Serial.println(topic);
```

```
String message;
for (unsigned int i = 0; i < length; i++) {
  message += (char)payload[i];
}

Serial.println("Raw payload:");
Serial.println(message);

StaticJsonDocument<256> doc;
DeserializationError error = deserializeJson(doc, message);

if (error) {
  Serial.print("JSON parse error: ");
  Serial.println(error.c_str());
  return;
}

const char* receivedKey = doc["key"];

if (receivedKey == nullptr || String(receivedKey) != DEVICE_KEY) {
  Serial.println("[ЗАХИСТ] Повідомлення відхилено: неправильний ключ вузла.");
  return;
}



int id = doc["PANEL_ID"];
if (id < 1 || id > 5) {
  Serial.println("Invalid PANEL_ID");
  return;
}

fleet[id].id = id;
fleet[id].temp = doc["temp"] | 0.0;
fleet[id].voltage = doc["vol"] | 0.0;
fleet[id].current = doc["cur"] | 0.0;
fleet[id].power = doc["pow"] | 0.0;
fleet[id].light = doc["light"] | 0;
fleet[id].active = true;

Serial.printf("Parsed panel %d | T=%.2f | V=%.2f | I=%.2f | P=%.2f | L=%d\n",
              fleet[id].id,
              fleet[id].temp,
              fleet[id].voltage,
```

```
                fleet[id].current,
                fleet[id].power,
                fleet[id].light);

  sendToDatabase(id);
  displayTable();
  analyzeAndControl();
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  for (int i = 0; i < 6; i++) {
    fleet[i].id = i;
    fleet[i].temp = 0;
    fleet[i].voltage = 0;
    fleet[i].current = 0;
    fleet[i].power = 0;
    fleet[i].light = 0;
    fleet[i].lastDbResponse = 0;
    fleet[i].active = false;
  }

  connectWiFi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  if (!client.connected()) {
    connectMQTT();
  }

  client.loop();
}
```

ДОДАТОК В - СХЕМА МОДЕЛЮВАННЯ СИСТЕМИ У СЕРЕДОВИЩІ WOKWI

Фрагмент схеми моделі у середовищі Wokwi

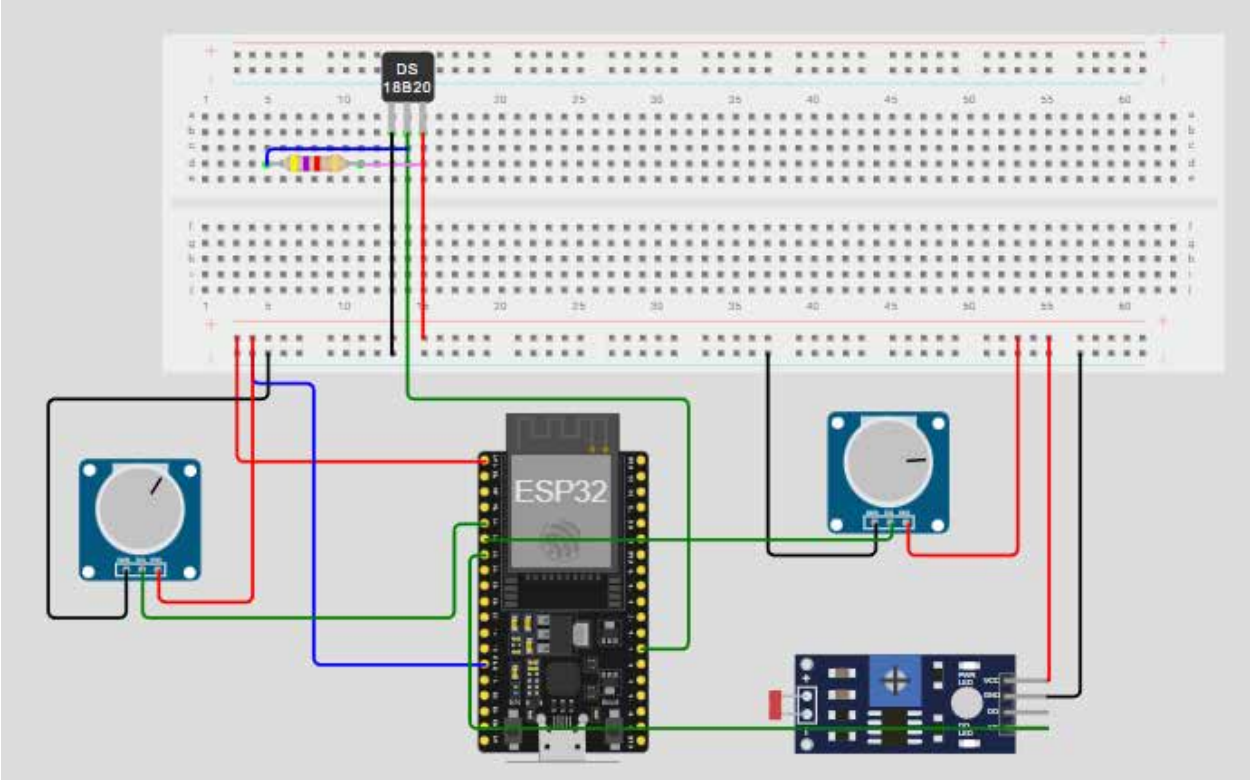

Рисунок В.1 - Загальна схема вузла моніторингу сонячної панелі у середовищі Wokwi

ДОДАТОК Г - SERIAL MONITOR, ТАБЛИЦІ, КОНСОЛЬНІ РЕЗУЛЬТАТИ

У даному додатку наведено результати роботи розробленої системи моніторингу, отримані у середовищі моделювання Wokwi за допомогою вбудованого інтерфейсу **Serial Monitor**. Наведені фрагменти підтверджують коректність функціонування як вузла сонячної панелі, так і контролера-шлюзу.

У процесі моделювання вузол панелі здійснює зчитування температури, освітленості, напруги та струму, після чого обчислює потужність і формує структуроване повідомлення у форматі JSON. Це повідомлення виводиться у Serial Monitor панелі та надсилається через MQTT до контролера-шлюзу.

```
{"panel_id": 1, "temp": 55.31, "vol": 9.87, "cur": 4.69, "pow": 46.32, "light": 75}
```

Рисунок Г.1 - Виведення JSON-повідомлень вузлом сонячної панелі у Serial Monitor

На рисунку Г.1 відображено приклад сформованих повідомлень, які містять ідентифікатор панелі, температуру, напругу, струм, потужність і рівень освітленості. Це підтверджує коректність роботи програмного модуля панелі та правильність формування пакета телеметричних даних.

Контролер-шлюз приймає повідомлення від вузлів панелей, виконує їх розбір, відображає поточний стан системи у вигляді таблиці та формує аналітичний висновок щодо режиму роботи сонячної електростанції.

```
| ID | Temp  | Volt  | Curr  | Power  | Light | DB Resp | Status |
|----|-------|-------|-------|--------|-------|---------|--------|
| 1  | 55.3  | 9.9   | 4.7   | 46.3   | 75  % | 200     | ACTIVE |

--- SES MESH INTELLIGENCE ---
Grid Power: 46.29 W | Weather (Sunlight): 75%
[СИСТЕМА] Баланс оптимальний.
------------------------------
```

Рисунок Г.2 - Виведення таблиці стану панелей і результатів аналізу на контролері-шлюзі

На рисунку Г.2 показано приклад табличного представлення параметрів активної панелі у Serial Monitor шлюзу. У таблиці наведено ідентифікатор панелі, температуру, напругу, струм, потужність, рівень освітленості, код відповіді сервера та статус активності. Після таблиці виводиться блок аналізу стану системи, у якому зазначаються сумарна потужність генерації, усереднений рівень освітленості та текстове повідомлення про поточний режим роботи.

```
ID | Temp | Volt | Curr | Power | Light | DB | Status
-------------------------------------------------------
 1 | 125.0 |  19.7 |   8.2 |  161.4 |    82 | 200 | OVERHEAT
 2 | 108.7 |  18.8 |   8.0 |  150.8 |    80 | 200 | OVERHEAT

--- SES MESH INTELLIGENCE ---
Grid Power: 312.13 W | Weather (Sunlight): 81%
[УВАГА] Сонячний пік! Можливе перевантаження інвертора.
[УВАГА] Панель 1 перегріта! Температура = 125.00 C
[УВАГА] Панель 2 перегріта! Температура = 108.69 C
------------------------------
```

Рисунок Г.3 - Повідомлення про сонячний пік та можливе перевантаження інвертора

Наведений результат показує, що контролер-шлюз аналізує не тільки параметри окремої панелі, а й сумарний стан усієї системи. Якщо рівень освітленості є високим, а загальна потужність перевищує встановлений поріг, система формує відповідне попередження.

Під час моделювання було встановлено, що при підключенні однієї панелі у таблиці відображається один рядок, а при збільшенні кількості активних вузлів таблиця автоматично розширюється по вертикалі. Це підтверджує працездатність механізму масштабування системи в межах закладеної програмної моделі.

Результати, наведені в додатку Г, підтверджують коректність формування телеметричних повідомлень, стабільність їх приймання контролером-шлюзом та правильність відображення й аналізу поточного стану системи моніторингу.

ДОДАТОК Д - WEBHOOK, HTTP POST, JSON

У даному додатку наведено результати перевірки передавання даних від контролера-шлюзу на серверну частину системи. Для тестування приймання HTTP POST-запитів у межах дипломної роботи було використано сервіс **webhook.site**, який дозволяє в режимі реального часу переглядати вміст отриманих мережевих запитів.

Після приймання MQTT-повідомлення від вузла сонячної панелі контролер-шлюз формує окремий JSON-документ, який містить основні параметри панелі та службову інформацію про її стан. Далі цей документ передається на сервер за допомогою HTTP POST-запиту.

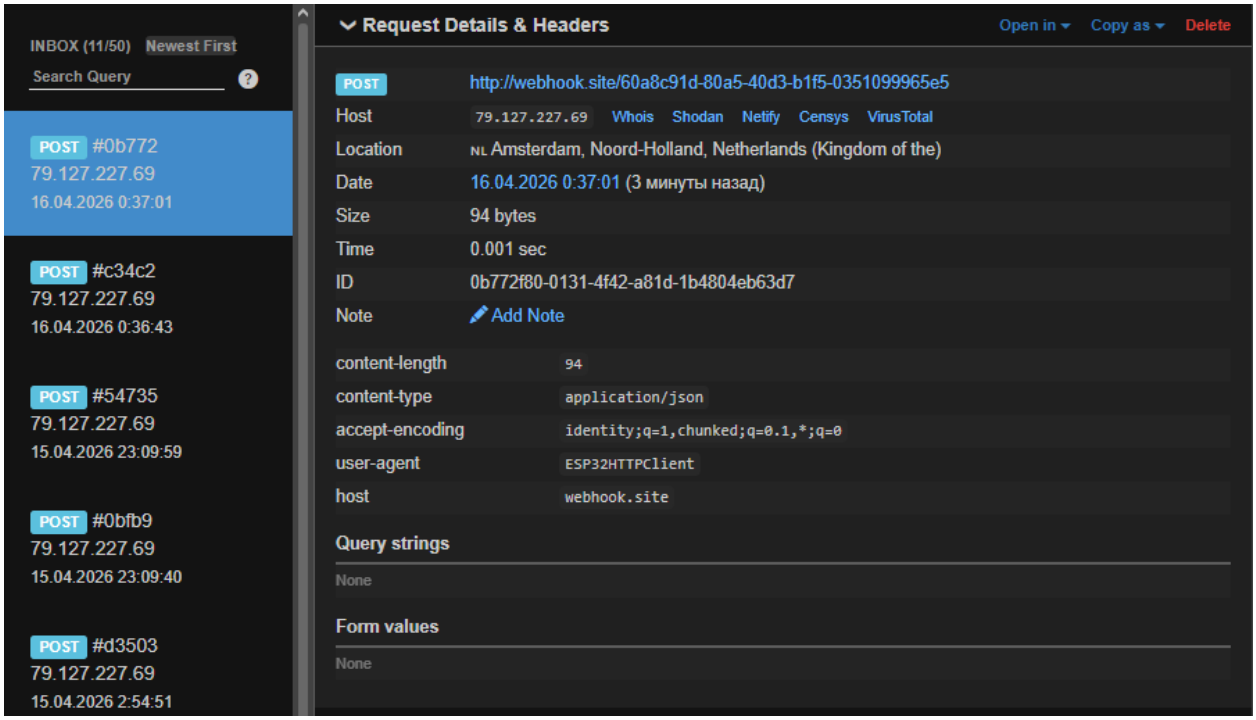

Рисунок Д.1 - Відображення HTTP POST-запиту на сервісі webhook.site

На рисунку Д.1 показано факт успішного отримання сервером HTTP POST-запиту від контролера-шлюзу. У вікні сервісу відображаються заголовки запиту, дата та час надсилання, тип вмісту `Application/json`, а також джерело надсилання. Це підтверджує, що шлюз коректно формує серверний запит і передає його у зовнішню систему.

У тілі запиту сервер отримує структуровані дані у форматі JSON, які містять основні параметри сонячної панелі.

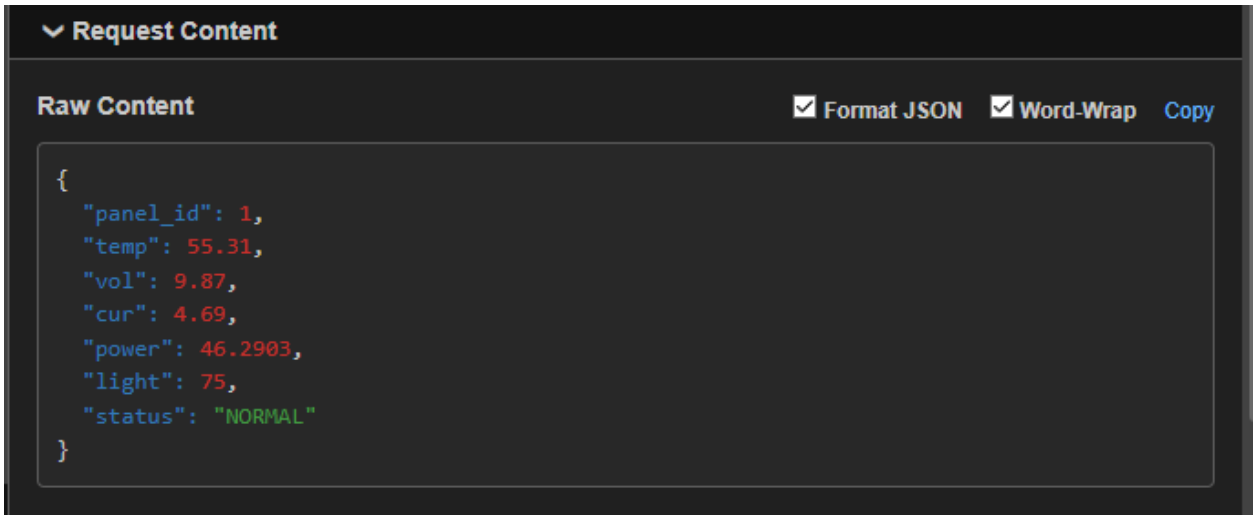

Рисунок Д.2 - Вміст JSON-повідомлення, отриманого сервером

На рисунку Д.2 наведено приклад JSON-документа, що був переданий шлюзом на сервер. У повідомленні містяться такі поля:

- `PANEL_ID` - ідентифікатор панелі;
- `temp` - температура панелі;
- `vol` - напруга;
- `cur` - струм;
- `power` - потужність;
- `light` - рівень освітленості;
- `status` - службова оцінка стану панелі.

Наявність поля status свідчить про те, що на сервер передаються не лише первинні телеметричні дані, а й результати локального аналізу, виконаного контролером-шлюзом. Це підвищує інформативність системи та дозволяє використовувати сервер не тільки для накопичення параметрів, але й для подальшого аналізу отриманих результатів.

Під час тестування було підтверджено, що сформований шлюзом JSON-документ коректно надходить до серверної частини, зберігає свою структуру та містить усі необхідні поля. Отже, результати, наведені у додатку Д, підтверджують працездатність заключного етапу системи, а саме передавання даних від контролера-шлюзу на сервер через HTTP.