Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Програмно-технічний засіб моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 022085.22.01.25 ПЗ

Виконала здобувачка IV курсу, група КІ2-22-1 _____ Софія СИРОТА
Підпис Ініціали, прізвище

Керівник _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«01» червня 2026 р. підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ПЕРШИЙ (БАКАЛАВРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сироті Софії Тарасівні
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-технічний засіб моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32

Керівник проекту (роботи) Лисенко Сергій Миколайович, д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз сучасних підходів до моніторингу та оптимізації енергоспоживання вузлів комп'ютерних мереж і постановка задачі щодо розробки програмно-технічного засобу моніторингу на базі мікроконтролера ESP32

Проєктування архітектури, структурної та принципової електричної схеми програмно-технічного засобу моніторингу енергоспоживання комп'ютерного обладнання

Розробка програмного забезпечення мікроконтролера ESP32 для збору, оброблення та передавання телеметричних даних, а також реалізація алгоритмів аварійного реагування і тестування системи у середовищі Wokwi

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Архітектура ПЗ проєкту

Архітектура ПЗ для кіберфізичної системи

Апаратне забезпечення проєкту

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня

7. Дата видачі завдання «_10_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітк
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконан
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконан
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2026	виконан
4	Робота над розділом 2 – вибір компонентів для проєктування системи адаптивного застосування моніторингових елементів розвідувального БПЛА	01.04.2026	виконан
5	Робота над розділом 3 – проєктування системи адаптивного застосування моніторингових елементів розвідувального БПЛА	29.04.2026	виконан
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконан
7	Попередній захист ВКР	26.05.2026	виконан
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач _____ Софія СИРОТА
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Сергій ЛИСЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 022085.22.01.25 ПЗ	Пояснювальна записка	73		
			Графічні матеріали			
2		КвРКІ 022085.22.01.25 ПЗ	Структурна схема	1		
			кіберфізичної системи			
3		КвРКІ 022085.22.01.25 ПЗ	Алгоритми роботи ПЗ	1		
			мікроконтролера			
4		КвРКІ 022085.22.01.25 ПЗ	Схема електрична	1		
			принципова			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 022085.22.01.25 ВП		
Розробив		Сирота			Відомість проекту	Літера	Аркуш / Аркушів
Перевір.		Лисенко				У	1 / 1
Н. контр.		Лисенко				ХНУ, КІ2-22-1	
Затв.		Павлова		01.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-апаратної система моніторингу параметрів комп'ютерного обладнання».

Автор роботи: Софія СИРОТА.

Керівник роботи: Сергій ЛИСЕНКО.

Пояснювальна записка: 73 с., 33 рис., 1 табл., 3 дод., 60 джерел.

Графічна частина: 3 креслення.

МІКРОКОНТРОЛЕР, МОНІТОРИНГ, MQTT, ПРОГРАМНО-ТЕХНІЧНИЙ ЗАСІБ, ESP32, ЕНЕРГОСПОЖИВАННЯ, КІБЕРФІЗИЧНА СИСТЕМА.

Кваліфікаційна робота бакалавра присвячена розробці та дослідженню програмно-технічного засобу моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32. Актуальність теми зумовлена зростанням енергоспоживання сучасної комп'ютерної інфраструктури, необхідністю підвищення енергоефективності серверного та мережевого обладнання, а також потребою у своєчасному виявленні аварійних режимів роботи електроживлення. Контроль параметрів електромережі у режимі реального часу дозволяє зменшити ризик перевантаження обладнання, попереджати пошкодження апаратних компонентів та забезпечувати стабільність функціонування вузлів комп'ютерної мережі.

Метою роботи є проєктування, реалізація та тестування програмно-апаратного комплексу для збору, оброблення та передавання телеметричних даних про енергоспоживання мережевого обладнання із можливістю локального аварійного реагування. Для досягнення поставленої мети було виконано аналіз сучасних підходів до побудови IoT-систем моніторингу, обрано апаратну платформу та вимірювальні модулі, розроблено структурну й принципову електричні схеми пристрою, створено алгоритми вимірювання параметрів електромережі та реалізовано програмне забезпечення мікроконтролера ESP32.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

Ефективна експлуатація сучасної ІТ-інфраструктури вимагає не лише нарощування обчислювальних потужностей, а й впровадження систем інтелектуального енергоменеджменту. В умовах високої щільності розміщення обладнання у центрах обробки даних (ЦОД) та корпоративних мережевих вузлах, витрати на електроенергію стають критичним фактором, що впливає на загальну економічну ефективність систем. Проте головною проблемою все ж залишається не лише вартість ресурсів, а й їх надійність. Так як неконтрольовані стрибки навантаження на блоки живлення та перегрів комутаційного обладнання є основними факторами апаратних відмов.

Більшість існуючих програмних рішень для моніторингу працюють на рівні операційних систем і не надають прямої інформації про фізичні параметри електричних кіл. Це створює так звані «сліпі зони» у діагностиці апаратного забезпечення. В свою чергу розробка локальних програмно-технічних засобів на базі мікроконтролерів дозволяє вирішити цю проблему шляхом впровадження концепції периферійних обчислень, коли збір та первинний аналіз телеметрії відбувається безпосередньо на рівні вузла споживання. Використання платформи ESP32 у цьому контексті є оптимальним вибором завдяки наявності двох ядер, інтегрованих Wi-Fi/Bluetooth модулів та розвиненої периферії для підключення аналогових сенсорів.

Науковою новизною отриманих результатів є вдосконалення методів технічної діагностики серверного обладнання шляхом поєднання апаратного моніторингу в реальному часі з алгоритмами автоматизованого реагування на основі аналізу профілів споживання.

Відповідно практичним значенням роботи є створення бюджетного та масштабованого рішення для ІТ-адміністраторів, яке дозволяє знизити ризики апаратних збоїв та оптимізувати витрати електроенергії без втручання в існуючу програмну архітектуру серверів.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАЛЬНОЇ ПРОБЛЕМИ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

Розвиток сучасної обчислювальної інфраструктури супроводжується постійним ускладненням архітектури постійним ускладненням архітектури мережевих вузлів, що, у свою чергу, ставить нові вимоги до систем електроживлення. На сьогоднішній день енергоспоживання комп'ютерного обладнання розглядається не лише як стаття операційних витрат, а й як основний чинник, що визначає технічний стан як життєвий цикл апаратних засобів. Аналіз експлуатації типових IT-інфраструктур (зокрема на прикладі ГО «IT кластер») дозволяє виокремити низку критичних проблем, що залишаються невирішеними в межах стандартних підходів до адміністрування [3].

Розглянемо проблему низької енергоефективності в режимах неповного навантаження. Одним із ключових аспектів є нелінійна залежність коефіцієнта корисної дії (ККД) імпульсивних блоків живлення від потужності споживання (рис. 1.1). Як продемонстровано на рис. 1.1, імпульсні блоки живлення, які стоять у кожному сервері чи комутаторі, мають одну неприємну особливість: вони працюють ефективно лише в тому випадку, коли завантажені на 50-80%. На практиці ж, більшість офісного та серверного обладнання значну частину часу перебуває в режимі очікування. Тобто фактично блок живлення працює на «обігрів всесвіту», втрачаючи величезну кількість енергії на власному опорі та внутрішніх процесах перетворення [4].

Коли ККД падає нижче 20%, це не лише зайві витрати коштів, а й зайве тепло. У закритих серверних шафах це тепло є основною загрозою, так як воно «висушує» електролітичні конденсатори в материнських платах, через що вони втрачають ємність, а система дає збої. Без детальної апаратної телеметрії адміністратор ніколи не дізнається, що сервер споживає забагато струму просто тому, що його блок живлення вже «підходить до кінця».

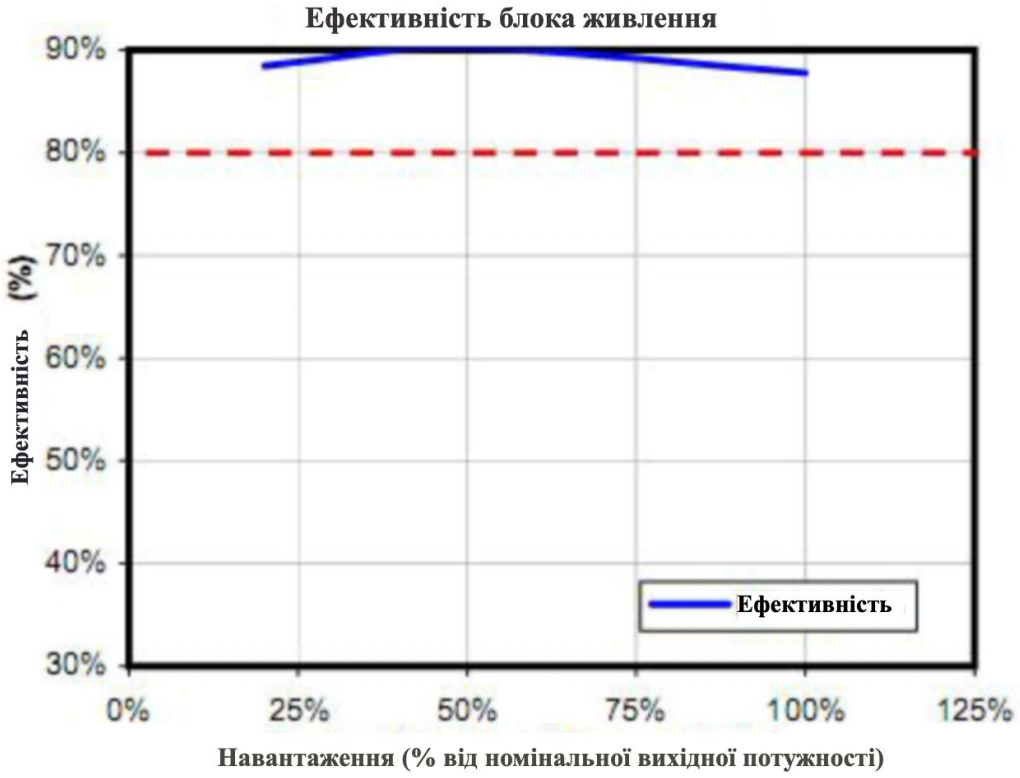


Рисунок 1.1 – Залежність коефіцієнта корисної дії (ККД) блока живлення від рівня навантаження

Значна частина обладнання функціонує в режимі очікування або неповного завантаження, де втрати на перетворення енергії є максимальними. Відсутність детальної телеметрії не дозволяє виявити такі «енергетичні аномалії» та прийняти рішення щодо перерозподілу навантаження або тимчасового вимкнення незадіяних вузлів [5].

Окрім фізичних аспектів споживання, важливо проаналізувати методи збору даних, що використовуються в сучасній інженерній практиці. Сьогодні домінують два основні підходи: програмний логінг та апаратне вимірювання. Логічна різниця між цими архітектурними рішеннями наочно продемонстрована на рис. 1.2.

Представлена схема демонструє, що програмний метод(праворуч) повністю залежить від програмного інтерфейсу (API) та стабільності операційної системи. У разі критичного збою системного софту, дані про енергоспоживання

стають недоступними. На противагу цьому, апаратний метод(ліворуч), який і покладено в основу даної роботи, передбачає використання незалежного вимірювального вузла. Це забезпечує об’єктивність телеметрії та можливість автономного управління живленням незалежно від стану основного обчислювального пристрою [6-7].

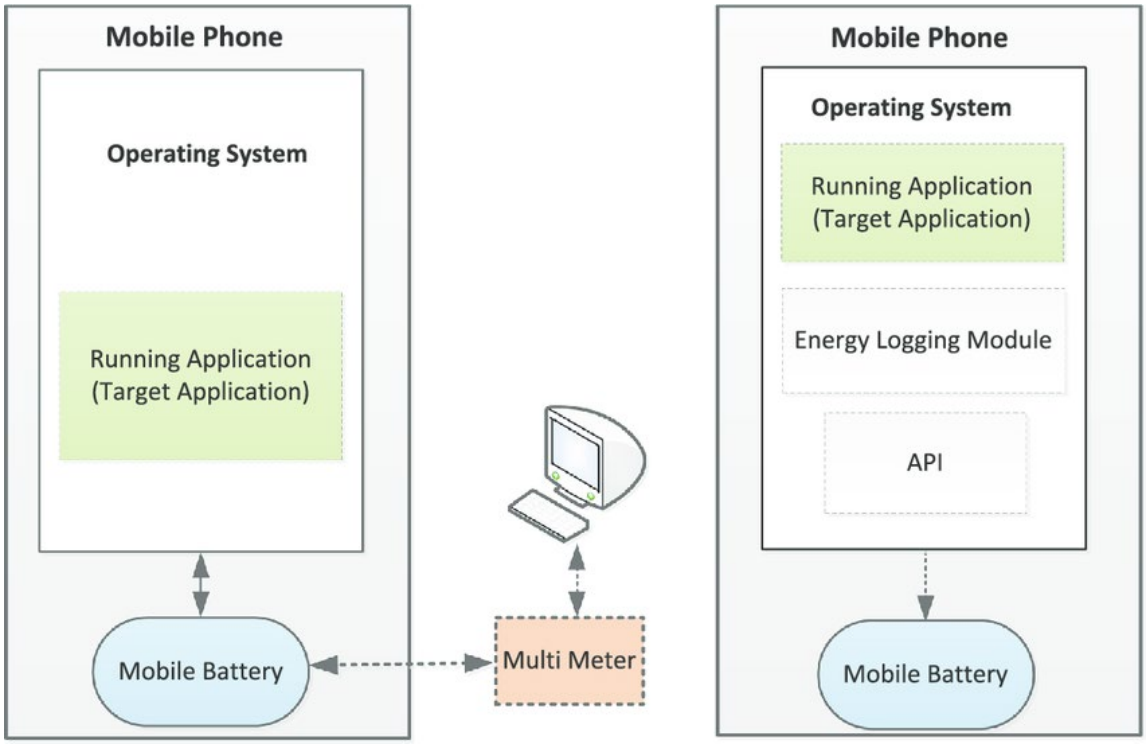


Рисунок 1.2 – Архітектурні моделі збору даних про енергоспоживання: апаратний(ліворуч) та програмний(праворуч) методи [**Error! Reference source not found.**]

Розглянемо наступний чинник – деградацію апаратної частини та ризики відмов. З точки зору комп’ютерної інженерії, особливе занепокоєння викликає теплова деградація компонентів. Збільшення споживаного струму часто є ознакою знову конденсаторів або погіршення властивостей термоінтерфейсів. Стандартні засоби моніторингу на рівні операційних систем фіксують лише відхилення в лініях живлення, що призводить до раптових апаратних збоїв без попередньої діагностики [8-10].

Також варто звернути увагу на недоліки існуючих інструментів контролю. Було проведено аналіз ринку та технічної літератури, і можна зробити висновок, що між програмним та апаратним моніторингом присутній суттєвий розрив.

По-перше, програмна залежність. Більшість систем моніторингу (наприклад, протокол SNMP - Simple Network Management Protocol) припиняють своє функціонування у разі критичного зависання системи або ж мережевого інтерфейсу [11].

По-друге, присутня обмеженість фізичних метрик. Існуючі засоби не дозволяють відстежувати якісні показники напруги та струму, такі як амплітудні стрибки або гармонічні спотворення, що виникають в умовах щільного розміщення обладнання.

По-третє, вартість інтеграції є досить високою. Промислові інтелектуальні системи розподілу живлення (Managed PDU) часто мають закриту архітектуру, що унеможливлює їх гнучку адаптацію під потреби конкретних невеликих організацій [13].

Тому враховуючи виявлені проблеми, виникає необхідність у створенні автономного програмно-технічного засобу, який би поєднував у собі функції вимірювального пристрою та інтелектуального контролера. Для досягнення мети роботи заплановано реалізувати наступні завдання:

1. Забезпечення високої точності дискретизації аналогових сигналів напруги та струму. Це дозволить отримувати «чисті» дані безпосередньо з фізичного рівня (датчиків) незалежно від обчислювального стану основного сервера.

2. Реалізація алгоритмів локальної обробки та аналітики даних на стороні мікроконтролера. Це передбачає впровадження математичних моделей для розрахунку активної та повної потужності безпосередньо вбудованим програмним забезпеченням, що розвантажує мережевий канал.

3. Організація енергоефективної мережевої диспетчеризації через протокол MQTT (Message Queuing Telemetry Transport). Завдання полягає у

створенні надійного стека передачі телеметричних даних, який дозволить інтегрувати пристрій у систему моніторингу з мінімальними накладними витратами на трафік [10].

4. Впровадження механізмів активного та дистанційного керування електроживленням. Це дозволить системі не лише знімати показники, а й оперативно реагувати на аварійні ситуації (стрибки напруги, перевантаження), виконуючи функцію інтелектуального запобіжника.

5. Забезпечення цілісності та синхронізації телеметричних потоків у часі. Використання мережевих протоколів синхронізації дозволить формувати точну історію енергоспоживання, що є необхідним для подальшого аналізу та оптимізації роботи мережевих вузлів [14-16].

1.2 Концепція вбудованих систем та їх класифікація

Вбудована система – це спеціалізована комп'ютерна система, яка інтегрована безпосередньо в пристрій, яким вона керує, або ж в інфраструктуру, яку вона моніторить. На відміну від персональних комп'ютерів загального призначення, вбудовані системи проектуються для виконання чітко визначеного набору функцій з жорсткими вимогами до надійності, мінімального енергоспоживання та реального часу виконання операцій. У даному випадку така система стає «інтелектуальним посередником» між електромережею та адміністратором [17].

Важливим пунктом є архітектурні особливості та безпосередньо роль у моніторингу вбудованих систем. Тому з погляду комп'ютерної інженерії, вбудовані системи є основою кіберфізичних систем, оскільки поєднують у собі цифрову логіку та фізичні процеси. У задачах моніторингу енергоспоживання їх головною перевагою є можливість безпосередньої взаємодії з аналоговим середовищем через вбудовані периферійні модулі. Зокрема аналогово-цифрові перетворювачі (АЦП) дозволяють «цифрувати» напругу (здійснювати

дискретизацію та квантування сигналу) та струм, таймери-лічильники забезпечують точні інтервали зняття відліків, а інтерфейси передачі даних відповідають за зв’язок із зовнішнім світом 18-20].

Використання вбудованих систем у даній роботі обґрунтовано необхідністю створення «автономного шару» моніторингу, який би функціонував незалежно від центральних процесорів серверного обладнання. Це дозволяє уникнути затримок, спричинених планувальником завдань операційної системи загального призначення. Це називається забезпеченням детермінованості обробки сигналів – тобто точно відомо, що сигнал буде оброблений у заданий момент часу, незалежно від навантаження на основний сервер [23].

Для правильного підбору апаратної бази необхідно класифікувати вбудовані системи за декількома ключовими ознаками. Класифікація вбудованих систем за обчислювальною потужністю та розрядністю дозволяє чітко розмежовувати сфери їх застосування залежно від складності розв’язуваних задач. Зокрема, до низькорівневих систем відносять 8-бітові та 16-бітові рішення, які найчастіше базуються на простих мікроконтролерах, таких як архітектура AVR . Попри свою економічну доцільність та низьку ціну, такі пристрої мають суттєві обмеження при реалізації складних математичних алгоритмів або розгортанні повноцінних мережевих протоколів [21].

Значно ширші можливості надають системи середнього рівня на базі 32-бітних архітектур, як-от ARM Cortex або Xtensa. Ця архітектура, що використовується в ESP32, дозволяє виконувати операції з плаваючою комою на апаратному рівні. А також саме цей клас є найбільш розповсюдженим в інженерній практиці, оскільки він забезпечує достатню продуктивність для проведення складних обчислень параметрів потужності в реальному часі при одночасній підтримці стеків TCP/IP. Найвищий рівень обчислювальної ієрархії займають високопродуктивні системи на кристалі (SoC), архітектура яких дозволяє функціонувати під управлінням полегшених операційних систем [22].

Аналізуючи вбудовані системи за функціональною складністю, їх прийнято розділяти на автономні та мережеві. Автономні рішення розраховані на виконання локальних операцій без необхідності зовнішньої комунікації, тоді як мережеві або IoT-орієнтовані системи оснащуються інтегрованими інтерфейсами зв'язку, серед яких найбільш поширеними є Wi-Fi, Bluetooth та Ethernet. В даній роботі об'єкт розробки належить саме до класу мережевих систем, що обумовлено необхідністю віддаленої передачі телеметричних даних.

Крім того, критично важливим аспектом є поділ систем за режимом роботи у реальному часу. У системах м'якого реального часу певна затримка в обробці інформації не вважається критичною для загального функціонування. Проте для задач технічної діагностики та захисту електромереж частіше виникає потреба у системах жорсткого реального часу. У такому випадку будь-яка несвоєчасна реакція на подію, наприклад на миттєвий стрибок напруги або ж струму, може призвести до незворотніх наслідків, а саме руйнування апаратної частини обчислювального вузла, що висуває особливі вимоги до швидкодії керуючого мікроконтролера [24-25].

У контексті ГО «ІТ кластер», вбудовані системи моніторингу виконують роль «інтелектуальних сенсорів». Завдяки використанню енергоефективних мікроконтролерів, такі системи можуть працювати роками, споживаючи мінімум енергії, при цьому забезпечуючи безперервний потік телеметричних даних до центральної бази даних або ж хмарного сервісу. Таким чином створюється фундамент для побудови проактивних систем захисту та енергоменеджменту [26].

1.3 Порівняльний аналіз апаратних платформ для розробки пристрою

Для реалізації програмно-технічного засобу моніторингу енергоспоживання необхідно обрати обчислювальну платформу, яка б збалансовано поєднувала продуктивність, наявність мережевих інтерфейсів та

низьку вартість. У сучасній комп'ютерній інженерії для подібних задач зазвичай найчастіше розглядаються три сімейства мікроконтролерних платформ: Arduino (архітектура AVR), STM32 (архітектура ARM) та ESP32 (архітектура Xtensa) [27-29].

Для більш наочного представлення технічних розбіжностей між обраними рішеннями та обґрунтування вибору, основні параметри платформ зведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика апаратних платформ

Характеристика	Arduino(Uno)	STM32 (F103C8)	ESP32(WROOM)
Архітектура	8-біт AVR	32-біт ARM Cortex-M3	32-біт Xtensa LX6
Тактова частота	16 МГц	72 МГц	160/240 МГц
Кількість ядер	1	1	2 (Dual Core)
RAM(оперативна)	2 КБ	20 КБ	520 КБ
Мережа (Wi-Fi)	Відсутня (зовнішня)	Відсутня (зовнішня)	Вбудована
АЦП(розрядність)	10 біт	12 біт	12 біт

Платформа Arduino (AVR) тривалий час була стандартом для швидкого прототипування завдяки простоті розробки та великої кількості бібліотек. Проте, з точки зору завдань даної роботи, 8-бітна архітектура AVR має суттєві недоліки. Обмежений обсяг оперативної пам'яті (2-8 КБ) та низька тактова частота(16 МГц) не дозволяють ефективно обробляти складні стеки мережевих протоколів та здійснювати високошвидкісну дискретизацію сигналів. Окрім того, відсутність вбудованих модулів бездротового зв'язку вимагає підключення зовнішніх шилдів, що збільшує габарити пристрою та відповідно знижує його надійність [30].

В свою чергу сімейство STM32 (ARM Cortex-M) є галузевим стандартом для промислових вбудованих систем. Ці контролери мають високу розрядність(32 біти), потужні багатоканальні АЦП та велику кількість апаратних інтерфейсів. STM32 ідеально підходить для задач точного вимірювання, проте більшість моделей початкового та середнього рівня не мають інтегрованих модулів Wi-Fi або ж Bluetooth. Створення мережевого пристрою на базі STM32 вимагає складної програмної реалізації мережевого рівня та використання додаткових контролерів зв’язку, що відповідно здорожує кінцеву розробку [31-33].

Платформа ESP32 є найбільш адаптованим рішенням для концепції Інтернету речей (IoT). На відміну від попередників, ESP32 базується на двоядерному 32-бітному процесорі Xtensa LX6, що дозволяє розділяти завдання збору даних та мережевої комунікації між різними ядрами. З погляду системного програмування вбудованих систем, наявність двох ядер (Core 0 та Core 1) в ESP32 дозволяє реалізувати детерміновану обробку даних. Таким чином можливо призначити для першого ядра високий пріоритет виконання задач зняття сигналу з АЦП, тоді як друге ядро буде зайняте обслуговуванням Wi-Fi стеку. Це вирішує проблему затримок, які часто виникають в одноядерних системах при одночасній роботі з мережею та датчиками.

Це забезпечує високу стабільність роботи системи: одне ядро може безперервно опитувати датчики струму, а інше відповідно підтримувати зв'язок з MQTT-брокером.

Головними перевагами ESP32 для даного проєкту є:

1. Наявність вбудованого датчика температури та сенсорів дотику, що можна використати для діагностики стану самого пристрою.

2. Інтегровані модулі Wi-Fi та Bluetooth, що дозволяють підключати пристрій до інфраструктури «ІТ кластеру» без прокладання додаткових кабелів.

3. Широкий динамічний діапазон АЙП та підтримка апаратного шифрування даних, що важливо для безпеки передачі комерційної інформації про енергоспоживання підприємства [35-27].

За результатами даного аналізу було виявлено, що ESP32 є оптимальним вибором, так як пропонує потужність промислового контролера STM32 у поєднанні з вбудованими мережевими можливостями, що є критичним для створення автономного засобу моніторингу.

1.4 Способи та протоколи передавання телеметричних даних у вбудованих системах

Організація обміну інформацією між вузлом моніторингу та центральною системою управління є однією з ключових задач комп'ютерної інженерії при проєктуванні розподілених систем. У контексті даної роботи вибір мережевих інтерфейсів та протоколів безпосередньо впливає на завантаження апаратних ресурсів мікроконтролера, стабільність роботи стека TCP/IP та загальну детермінованість системи [38].

Основним фізичним рівнем (за моделлю OSI) для передачі даних обрано стандарт бездротового зв'язку IEEE 802.11 (Wi-Fi). Вибір даного інтерфейсу обґрунтований наявністю інтегрованого в архітектуру мікроконтролера ESP32 радіочастотного (RF) тракту та окремого апаратного прискорювача для шифрування даних (WPA2/WPA3). Це дозволяє винести мережеву обробку на низький рівень, звільняючи основні обчислювальні ядра для вимірювання фізичних параметрів енергомережі. Використання Wi-Fi дозволяє інтегрувати пристрій у наявну інфраструктуру «IT кластеру» без модернізації кабельної мережі, що спрощує розгортання системи.

Проте на вищих рівнях моделі OSI виникає потреба в оптимізації обміну даними для мінімізації накладних витрат (overhead) на формування мережевих пакетів. Аналіз показує, що використання протоколу HTTP є недоцільним для

вбудованих систем через великий розмір текстових заголовків та необхідність постійного перепідключення, що виснажує SRAM мікроконтролера.

Натомість найбільш ефективним рішенням для передачі телеметрії є протокол MQTT (Message Queuing Telemetry Transport). З точки зору архітектури вбудованих систем, його головна перевага полягає у використанні бінарного формату заголовків, мінімальний розмір яких становить лише 2 байти. Це значно знижує навантаження на обчислювальне ядро та звільняє ресурси оперативної пам'яті (SRAM) для виконання критично важливих завдань – зняття та обробки сигналів у реальному часі [39].

На представленій діаграмі зображена архітектурна модель розподіленої системи збору та аналізу даних, де центральним вузлом виступає MQTT Broker. У контексті даної роботи мікроконтролери ESP32 виконують роль Publishers (Видавців), які передають первинну телеметрію про стан енергомережі.

Інтелектуальний брокер включає в себе три важливих компоненти. По-перше, Fuzzy module (Модуль нечіткої логіки), що складається з етапів фазифікації, логічного виведення та дефізифікації. Цей блок дозволяє системі приймати «гнучкі» рішення (наприклад, оцінювати рівень критичності перевантаження) та генерувати сигнал Decision для автоматичного управління реле [40-42].

По-друге, Network Analyzer (Мережевий аналізатор) відповідає за збір статистики трафіку та навчання системи на основі сформованих наборів даних. Це забезпечує оптимізацію завантаження каналів зв'язку Wi-Fi.

По-третє, Підписники, тобто вузли, що отримують оброблену інформацію. Це можуть бути робочі станції адміністраторів IT-кластеру або ж автоматизовані бази даних, як приклад [43]. Архітектурну схему інтелеуктуальної системи моніторингу на основі протоколу MQTT можна побачити на рисунку 1.3.

Принциповою особливістю MQTT є модель взаємодії «видавець-підписник», що дозволяє реалізувати асинхронний обмін даними. В умовах IT-інфраструктури це означає, що мікроконтролер не повинен підтримувати

постійний активний сеанс зв’язку з сервером, а може відправляти дані лише при зміні показників споживання (наприклад, при стрибку напруги). Крім того, наявність механізмів QoS(Quality of Service) дозволяє інженеру програмно визначити рівень гарантії доставки пакетів: від мінімального завантаження каналу до обов’язкового підтвердження отримання критичної аварійної інформації.

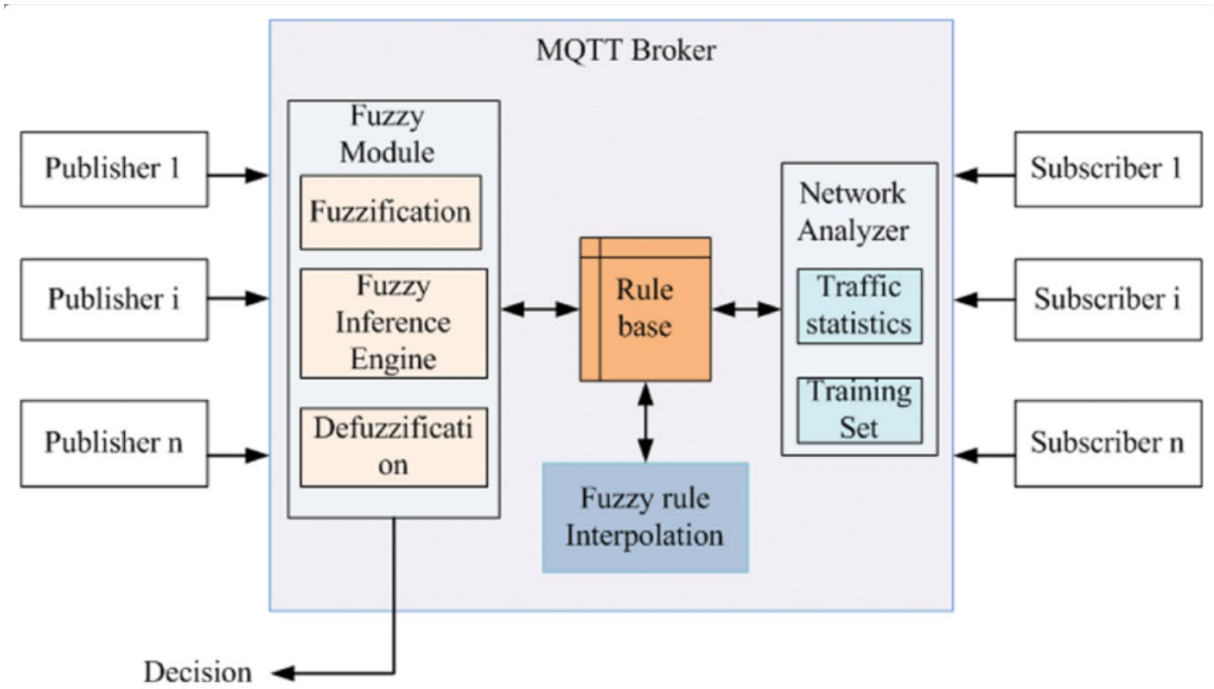


Рисунок 1.3 – Архітектурна схема інтелектуальної системи моніторингу на основі протоколу MQTT з використанням модулів нечіткої логіки та аналізу мережевого трафіку [**Error! Reference source not found.**]

Вибір правильного рівня QoS дозволяє збалансувати навантаження на мережу та гарантувати доставку критичних даних. Ми можемо використовувати три рівні: QoS 0, QoS 1 та QoS 2 [44-45].

Рівень QoS 0 (At most once) передбачає, що повідомлення відправляється один раз без підтвердження. Це ідеально для регулярної статистики (наприклад, кожні 10 секунд), де втрата одного пакета не є критичною.

Рівень QoS 1 (At least once) гарантує, що повідомлення дійде до брокера, але воно може дублюватися. Це підходить для загальних звітів про споживання.

Рівень QoS 2 (Exactly once) є найскладнішим рівнем, який гарантує доставку рівно один раз. Це критично важливо для команд управління реле або сповіщень про аварії, щоб уникнути помилкових повторень спрацювань системи захисту.

Окрім цього протокол MQTT підтримує функцію Keep live та механізм Last Will and Testament (LWT). У контексті моніторингу вузлів IT-кластеру це може працювати наступним чином: якщо мікроконтролер ESP32 раптово втратить живлення або зв'язок через аварію, брокер автоматично розішле всім підписникам(адміністраторам) заздалегідь підготовлене «заповітне» повідомлення про те, що вузол пішов у офлайн. Це дозволяє реалізувати проактивну діагностику мережі без постійного запитування (polling) кожного пристрою з боку сервера, що суттєво розвантажує обчислювальні потужності всієї системи [46].

Поєднання Wi-Fi як фізичного каналу та MQTT як прикладного протоколу створює свого роду гнучкий та надійний мережевий стек, який є адаптованим до обмежених ресурсів мікроконтролерних систем.

1.5 Постановка задачі

Метою даної роботи є проєктування та реалізація програмно-технічного засобу для автоматизованого моніторингу та інтелектуального керування енергоспоживання вузлів комп'ютерної мережі. В основі рішення лежить використання 32-бітного мікроконтролера ESP32, який має забезпечити поєднання функцій збору аналогових даних, обчислювальної обробки та мережевої комунікації.

Згідно з визначеною метою, на розробку системи покладаються наступні технічні та функціональні завдання.

По-перше, проєктування апаратного модуля збору даних. Розробка схеми підключення сенсорів струму та напруги до аналогово-цифрового перетворювача (АЦП) мікроконтролера. Система повинна забезпечувати високу частоту дискретизації сигналу для отримання точної форми хвилі змінного струму, що є необхідним для технічної діагностики стану імпульсних блоків живлення серверного обладнання.

По-друге, реалізація математичного апарату обчислень. Програмне забезпечення мікроконтролера повинно виконувати розрахунок миттєвої та середньоквадратичної (RMS) потужності у реальному часі. Окрім поточного споживання у Ватах, система має здійснювати інтегрування показників для накопичення статистичних даних про енерговитрати (у кВт на год) за визначені проміжки часу.

По-третє, розробка мережевого інтерфейсу передачі даних. Використання вбудованого Wi-Fi стеку ESP32 для забезпечення передачі телеметричної інформації на центральний сервер або хмарну платформу. Обмін даними має здійснюватися за допомогою легкого протоколу MQTT, що дозволить мінімізувати навантаження на корпоративну мережу та забезпечити швидке оновлення графіків моніторингу на терміналі адміністратора [47-50].

По-четверте, створення контуру інтелектуального керування та захисту. Реалізація алгоритмів автоматичного керування живленням через модуль реле. Система повинна функціонувати у двох режимах. Перший режим – режим оптимізації, автоматичне відключення вузлів, що тривалий час перебувають у стані простою, для зменшення енергетичних втрат. Другий режим – режим «розумного запобіжника», автономне реагування на критичні аварійні ситуації (стрибки напруги, перевантаження), що передбачає миттєве відключення навантаження без очікування команди від центрального сервера.

По-п’яте, забезпечення відмовостійкості та автономності. Логіка безпеки повинна бути реалізована на рівні вбудованого ПЗ мікроконтролера, що гарантує

працездатність захисних функцій навіть у разі втрати мережевого з’єднання або збою центральної системи моніторингу.

1.6 Висновки до першого розділу

У першому розділі було проведено аналіз проблематики моніторингу енергоспоживання вузлів комп’ютерних мереж та досліджено сучасні підходи до побудови вбудованих систем контролю електроживлення. Встановлено, що традиційні програмні засоби моніторингу не забезпечують прямого доступу до фізичних параметрів електромережі та залежать від стабільності операційної системи серверного обладнання. Це створює ризики виникнення «сліпих зон» у процесі технічної діагностики та не дозволяє своєчасно реагувати на аварійні ситуації.

У процесі дослідження було проаналізовано архітектурні особливості вбудованих систем, принципи їх функціонування та класифікацію за обчислювальною потужністю, типом взаємодії з мережею та режимом реального часу. Визначено, що для задач моніторингу параметрів електроживлення найбільш доцільним є використання мережевих 32-бітних мікроконтролерних платформ із підтримкою бездротової передачі даних та багатоканального аналого-цифрового перетворення.

Проведений порівняльний аналіз платформ Arduino Uno, STM32 та ESP32 показав, що саме ESP32 найбільш повно відповідає вимогам розроблюваної системи. Основними перевагами платформи є висока обчислювальна продуктивність, наявність інтегрованих модулів Wi-Fi та Bluetooth, підтримка багатоканального 12-бітного АЦП, а також можливість одночасного виконання задач збору даних та мережевої взаємодії.

Також у розділі було досліджено особливості організації передачі телеметричних даних у вбудованих системах. Встановлено, що використання протоколу MQTT у поєднанні з бездротовим інтерфейсом Wi-Fi дозволяє

створити енергоефективну та масштабовану систему обміну даними з мінімальними накладними витратами на мережевий трафік.

За результатами проведеного аналізу сформульовано основні вимоги до програмно-технічного засобу моніторингу та поставлено задачі подальшого проєктування апаратної й програмної частин системи.

2 АПАРАТНА ТА ПРОГРАМНА БАЗА ПРОЄКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ НА БАЗІ ESP32

2.1. Опис мікроконтролера та залученого комутаційного обладнання

Розробка будь-якого вбудованого програмно-технічного засобу моніторингу енергоспоживання вузлів з комп'ютерної мережі вимагає ґрунтовного підходу до вибору елементної бази та інструментарію, а також інтеграції високовольтної силової електроніки з прецизійними (точними) цифровими обчислювальними системами.

У цьому розділі розглянуто технічну основу проєкту: мікроконтролер ESP32, вимірювальні та комутаційні модулі, систему живлення, середовище розробки й моделювання, а також програмні підходи, використані для реалізації пристрою моніторингу електромережі.

Проєктування апаратної частини системи моніторингу розпочинається з вибору мікроконтролера, який виконує роль центрального обчислювального вузла. Саме він забезпечує зчитування сигналів із датчиків, обробку виміряних значень, виконання математичних розрахунків та передачу даних через бездротові інтерфейси.

Оскільки система повинна працювати в режимі реального часу та одночасно виконувати кілька задач, до мікроконтролера висуваються підвищені вимоги щодо продуктивності, швидкодії аналого-цифрового перетворення та підтримки мережевих протоколів.

Для вибору оптимальної платформи було проведено порівняльний аналіз трьох популярних рішень для систем Інтернету речей. Це 8-бітний мікроконтролер ANmega328P (платформа Arduino Uno), 32-бітний одноядерний ESP8266 (NodeMCU) та 32-бітний двоядерний ESP32-WROOM-32. Порівняльну характеристику даних мікроконтролерів можна побачити в таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика мікроконтролерних платформ для систем моніторингу енергоспоживання

Характеристика	ATmega328P (Arduino Uno)	ESP8266 (NodeMCU)	ESP32 (ESP-WROOM-32)
Архітектура	8-біт AVR	32-біт Tensilica L106	32-біт Xtensa Dual-Core LX6
Тактова частота	16Мгц	80/160 Мгц	160/240 МГц
Оперативна пам’ять (SRAM)	2Кб	50Кб	520 КБ
АЦП (Роздільна здатність)	10-біт (1024 рівні)	10-біт (1 канал)	12-біт (4096 рівнів), 18 каналів
Бездротові інтерфейси	Відсутні	Wi-Fi 802.11 b/g/n	Wi-Fi, Bluetooth v4.2 BR/EDR та BLE
Підтримка багатопотоковості	Ні	Програмна імітація	Так

Аналіз характеристик показує, що система електромережі потребує значних обчислювальних ресурсів, особливо від час розрахунку середньоквадратичних значень струму та напруги в режимі реального часу. Додаткове навантаження створює необхідність високої частоти дискретизації сигналів та паралельної передачі даних мережею.

Платформа Arduino Uno на базі мікроконтролера ATmega328P має недостатню продуктивність для реалізації таких задач [60]. Основними обмеженнями є невеликий обсяг оперативної пам’яті, низька тактова частота, відсутність вбудованих бездротових інтерфейсів та 10-бітний аналого-цифровий перетворювач, точності якого недостатньо для високоточних вимірювань параметрів електромережі.

Платформа ESP8266 має значно кращі мережеві можливості та вищу обчислювальну продуктивність, однак її апаратним недоліком є наявність лише одного аналогового входу. Це ускладнює одночасне вимірювання струму та напруги без використання додаткових зовнішніх мультиплексорів.

З урахуванням наведених характеристик оптимальним рішенням для реалізації системи було обрано мікроконтролер ESP32 модифікації WROOM-32. Дана платформа поєднує високу обчислювальну потужність, багатоканальний 12-бітний АЦП, підтримку Wi-Fi та Bluetooth.

Висока обчислювальна потужність мікроконтролера ESP32 досягається завдяки використанню гарвардської архітектури на базі двоядерного процесора Xtensa LX6, розробленого компанією Tensilica. На відміну від класичної архітектури фон Неймана, у якій для передачі інструкцій та даних використовується спільна системна шина, гарвардська архітектура передбачає фізичне розділення шин пам’яті програм та пам’яті даних. Подібний підхід дозволяє обчислювальному ядру одночасно виконувати зчитування машинних інструкцій та операції доступу до даних у межах одного машинного такту, що суттєво підвищує загальну продуктивність системи та зменшує затримки при виконанні паралельних обчислювальних операцій

Тактова частота процесорних ядер ESP32 може динамічно змінюватися в діапазоні від 160 МГц до 240 МГц залежно від поточного навантаження та енергетичного режиму роботи системи. За рахунок цього мікроконтролер здатний забезпечувати обчислювальну продуктивність на рівні до 600 DMIPS (Dhrystone Million Instructions Per Second), що суттєво перевищує можливості традиційних 8-бітних мікроконтролерів AVR-архітектури. Наявність двох незалежних процесорних ядер (Core 0 та Core 1) дозволяє реалізувати багатозадачний режим роботи на апаратному рівні. У рамках розроблюваної системи одне ядро може бути виділене для обслуговування переривань, роботи аналого-цифрового перетворювача та виконання складних математичних обчислень, пов’язаних із визначенням параметрів електромережі, тоді як друге

ядро використовується для підтримки бездротового стеку Wi-Fi/Bluetooth, мережевих протоколів та обміну даними із серверною частиною системи моніторингу.

Додатковою перевагою мікроконтролера ESP32 є наявність апаратного блоку обчислень із рухомою комою FPU (Floating Point Unit). Інтеграція FPU дозволяє виконувати складні арифметичні операції безпосередньо на апаратному рівні, зокрема операції ділення, множення та добування квадратного кореня, що активно використовуються під час обчислення середньоквадратичних значень струму та напруги (RMS). Виконання таких операцій займає лише кілька машинних тактів, тоді як у 8-бітних мікроконтролерах аналогічні обчислення реалізуються програмною емуляцією та можуть вимагати сотень машинних тактів процесора. Це дозволяє значно зменшити навантаження на центральне ядро та забезпечити виконання математичних розрахунків у режимі реального часу без втрати точності вимірювання.

Окремою інженерною перевагою платформи ESP32 є наявність співпроцесора наднизького енергоспоживання ULP (Ultra Low Power Coprocessor), який побудований на спрощеній RISC-архітектурі. Основним призначенням ULP є виконання елементарних операцій моніторингу та збору даних у режимах мінімального енергоспоживання. Навіть у випадку переходу основних процесорних ядер у режим глибокого сну співпроцесор здатний продовжувати періодичне опитування каналів аналого-цифрового перетворювача, споживаючи при цьому струм на рівні кількох мікроампер. Хоча у розробленій системі моніторингу основне навантаження покладається саме на високопродуктивні ядра Xtensa LX6, наявність ULP відкриває можливість подальшої модернізації пристрою та реалізації енергоефективних режимів функціонування в автономних системах контролю.

Організація пам’яті мікроконтролера ESP32 включає 520 КБ внутрішньої статичної оперативної пам’яті SRAM, яка використовується для зберігання програмних даних та буферів. Окрім цього, мікроконтролер містить 448 КБ

постійної пам’яті ROM, призначеної для зберігання завантажувача, базових системних бібліотек та низькорівневих функцій ядра. Архітектура ESP32 також підтримує підключення зовнішньої SPI Flash-пам’яті обсягом до 16 МБ, що використовується для зберігання прошивки, файлової системи, конфігураційних параметрів та мережевих даних системи моніторингу.

Особливої уваги заслуговує підсистема аналого-цифрового перетворення. ESP32 оснащений двома багатоканальними 12-бітними АЦП регістру послідовних наближень (SAR ADC), що підтримують до 18 каналів вимірювання. Роздільна здатність у 12 біт забезпечує 4096 рівнів квантування ($2^{12} = 4096$), що забезпечує можливість реєстрації малих змін напруги на виходах первинних сенсорів. Однак, використання АЦП ESP32 вимагає врахування його апаратних особливостей. Зокрема, на краях діапазону вимірювання (нижче 0.1 В та вище 3.1 В) спостерігається нелінійність характеристик перетворення, що призводить до появи нелінійних ділянок характеристики перетворення, відомих також як «мертві зони». Для компенсації цього на апаратному рівні сигнал від датчиків масштабується операційними підсилювачами таким чином, щоб робоча амплітуда синусоїди знаходилася у найбільш лінійній зоні АЦП (від 0.2 В до 3.0 В) , а внутрішній атенюатор налаштовується на рівень 11 дБ.

Для забезпечення коректного відновлення форми синусоїдального сигналу під час цифрової обробки частота дискретизації аналого-цифрового перетворення повинна щонайменше вдвічі перевищувати частоту мережі відповідно до теореми Котельникова–Шеннона. Оскільки частота промислової електромережі становить 50 Гц, мінімально допустима частота дискретизації перевищує 100 Гц. Проте для точного обчислення середньоквадратичних значень, активної потужності та аналізу форми сигналу у системі доцільно використовувати частоту дискретизації в межах 1–5 кГц, що дозволяє отримати достатню кількість вибірок на період синусоїди та зменшити похибку цифрових обчислень.

Для зменшення впливу високочастотних шумів та паразитних електромагнітних завад перед входами АЦП може використовуватися RC-фільтр нижніх частот (anti-aliasing filter). Такий фільтр обмежує спектр вхідного сигналу та запобігає виникненню ефекту накладання спектрів під час дискретизації аналогового сигналу.

Для забезпечення взаємодії обчислювального ядра з навколишнім середовищем та периферійними пристроями, плата розробника ESP32 оснащена розширеною системою контактних виводів (Pinout). Особливістю архітектури Xtensa є наявність внутрішнього апаратного мультиплексора (GPIO Matrix). Цей механізм дозволяє програмно перепризначати більшість цифрових інтерфейсів (таких як I2C, SPI, UART, ШІМ) на будь-який зручний для розробника вивід, що спрощує процес трасування друкованої плати та оптимізацію структури підключень. Зовнішній вигляд плати ESP32 можна побачити на рисунку 2.1.

Рисунок 2.1 – Зовнішній вигляд та структурне розташування виводів плати розробника ESP32

Логічна організація контактних виводів мікроконтролера ESP32 передбачає їхній розподіл на чотири основні функціональні групи, кожна з яких виконує специфічні завдання в межах загальної архітектури пристрою. Першу групу складають виводи живлення, серед яких ключову роль відіграє контакт 5V (VIN), призначений для подачі вхідної нерегульованої напруги в межах від 5 В до 12 В. У структурі розробленої системи саме до цього виводу підключається вихідна лінія 5 В від імпульсного перетворювача HLK-PM01. Поруч із ним розташований вивід 3V3, що є виходом інтегрованого лінійного LDO-стабілізатора AMS1117-3.3 і забезпечує живлення внутрішньої логіки обчислювального ядра, тоді як контакти GND формують спільну шину заземлення для створення нульового потенціалу в усій схемі.

Наступна група представлена аналоговими входами багатоканального 12-бітного аналого-цифрового перетворювача, до яких належать контакти з маркуванням від D32 до D39, включаючи спеціалізовані піни VP та VN. Важливою інженерною особливістю виводів 34, 35, 36 та 39 є їхня здатність працювати виключно в режимі входу без використання внутрішніх підтягуючих резисторів, що робить їх оптимальними для прецизійних метрологічних вимірювань. У даному проєкті аналоговий сигнал від датчика струму ACS712 через резистивний дільник подається саме на вивід D34, тоді як масштабований сигнал від трансформатора напруги ZMPT101B зчитується суміжним каналом АЦП.

Цифрові інтерфейси вводу/виводу загального призначення (GPIO) охоплюють більшість доступних контактів, таких як 2, 4, 18, 23 та інші, забезпечуючи гнучкість налаштування як на вхід, так і на вихід. Слід враховувати, що ці виводи толерантні виключно до напруги 3.3 В, а перевищення цього рівня може призвести до незворотного пробою напівпровідникової структури кристала. Для реалізації логіки керування виконавчим механізмом у роботі використовується цифровий вивід D23, який програмно конфігурується

як вихід для генерації керуючого сигналу на оптопару EL817 модуля реле при ідентифікації критичного перевантаження.

Остання група включає службові та системні виводи, серед яких виділяється контакт EN, що відповідає за апаратне увімкнення та скидання чипа. Для забезпечення стабільного процесу завантаження ця лінія підтягується до живлення та з'єднується із заземленням через конденсатор, формуючи RC-ланцюг затримки. Додатково архітектура передбачає наявність так званих «Strapping pins» (GPIO 0, 2, 5, 12, 15), стан яких у момент старту визначає режим завантаження процесора з флеш-пам'яті або перехід у режим прошивки через UART. При конструюванні периферійних підключень ці піни вимагають особливої уваги для запобігання випадковому блокуванню коректного старту операційної системи пристрою.

Важливим критерієм вибору мікроконтролера ESP32 є його висока гнучкість у питаннях апаратної взаємодії з цифровою периферією. Окрім використання АЦП для зчитування аналогових сигналів, архітектура мікроконтролера містить апаратні контролери для підтримки промислових стандартів передачі даних: I2C, SPI та UART. Інтерфейс I2C (Inter-Integrated Circuit) в ESP32 представлений двома незалежними апаратними шинами.

У контексті систем моніторингу наявність I2C є критично важливою для майбутнього масштабування пристрою: ця шина дозволяє підключати зовнішні прецизійні модулі розширення (наприклад, зовнішні 16-бітні АЦП ADS1115 або OLED-дисплеї для локальної візуалізації даних) використовуючи всього дві лінії зв'язку - лінію даних (SDA) та лінію тактування (SCL). Апаратний контролер I2C в ESP32 підтримує роботу в стандартному (100 кбіт/с) та швидкому (400 кбіт/с) режимах, автоматично генеруючи стартові та стопові умови протоколу без навантаження на обчислювальне ядро.

Інтерфейс SPI в ESP32 реалізований трьома незалежними апаратними контролерами SPI, HSPI, VSPI. Використання шини SPI є необхідним у випадках, коли пристрій моніторингу потребує локального збереження великих масивів

телеметрії на зовнішню SD-карту у випадку тривалої втрати підключення до Wi-Fi мережі (режим дата-логера).

Завдяки технології прямого доступу до пам'яті (DMA - Direct Memory Access), контролер SPI здатний передавати буфери даних між периферією та оперативною пам'яттю (SRAM) мікроконтролера повністю автономно, звільняючи процесор для розрахунку параметрів RMS та активної потужності. Також чип обладнаний трьома апаратними модулями UART (Universal Asynchronous Receiver-Transmitter). UART0 традиційно резервується для прошивки та відлагодження програми через USB-міст, тоді як UART1 та UART2 можуть бути використані для підключення промислових лічильників електроенергії за протоколом Modbus RTU (через конвертер RS-485), що перетворює розроблений пристрій на універсальний шлюз збору даних промислового рівня.

Система моніторингу взаємодіє з мережею змінного струму високої напруги (220 В), що висуває жорсткі вимоги до електробезпеки. Тому головним критерієм вибору сенсорів є наявність гальванічної розв’язки між високовольтною та низьковольтною частинами пристрою.

Для вимірювання струму навантаження обрано мікросхему ACS712-20A. Принцип її дії базується на ефекті Холла: струм створює магнітне поле, яке фіксується кристалом Холла і перетворюється на пропорційну напругу. Такий метод дозволяє повністю ізолювати вимірювальну лінію (напруга ізоляції до 2.1 кВ). Вихідна напруга розраховується за формулою:

$$V_{out} = \frac{V_{CC}}{2} + (I * S), \qquad (2.1)$$

де V_{CC} – напруга живлення датчика (5В);

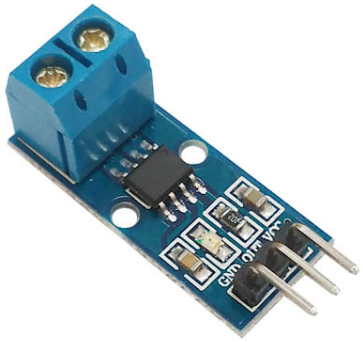
I – вимірюваний струм;

S – чутливість датчика (для моделі 20 А вона становить 100 мВ/А).

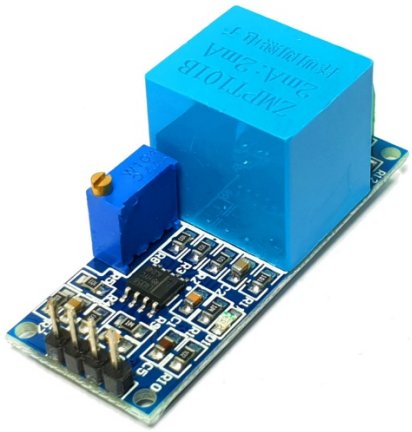
Контроль напруги здійснюється активним трансформаторним модулем ZMPT101B. Він гарантує високу лінійність передачі фази синусоїди та ізоляцію до 4000 В. Інтегрований операційний підсилювач LM358 гнучко налаштовує амплітуду вихідного сигналу під вимоги АЦП мікроконтролера. Його індуктивний характер може вносити фазовий зсув (Phase Shift), що вимагає програмної корекції масивів вибірок.

Для комутації навантаження задіяно електромеханічне реле SRD-05VDC-SL-C. Вузол керування реалізується через транзисторний ключ (наприклад, 2N2222) з оптоізоляцією EL817. Паралельно котушці реле обов'язково підключається зворотний діод 1N4007 для шунтування ЕРС самоіндукції.

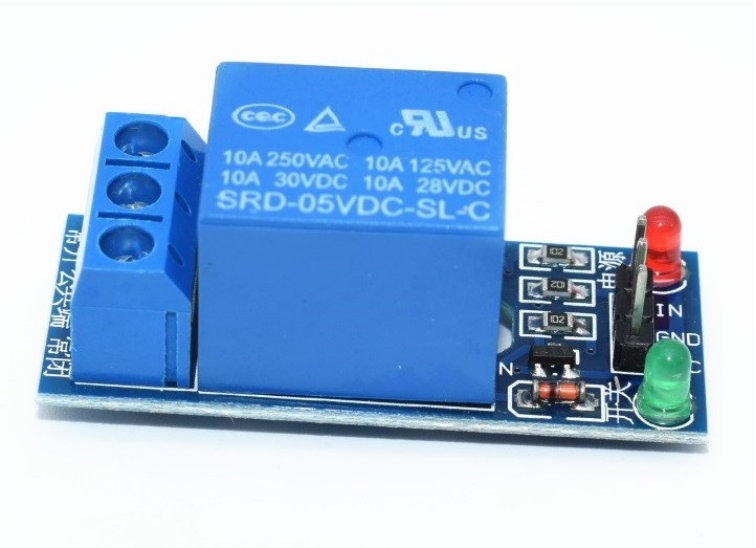
Система живлення побудована на базі імпульсного модуля HLK-PM01 (AC-DC 220В-5В) та LDO-регулятора AMS1117-3.3. Аналіз енергетичного балансу показує, що ESP32 (до 240 мА), реле (70 мА), ACS712 (10 мА), ZMPT101B (5 мА) та стабілізатор (20 мА) сумарно споживають $I_{max} = 345$ мА. Максимальна потужність становить P_{dev} = 5·0.345=1.725 Вт. При номінальній потужності джерела у 3 Вт коефіцієнт завантаження становить 57.5%, що гарантує стабільність роботи без просідань напруги.



а)



б)



в)

Рис. 2.2. Вимірювальні та комутаційні компоненти системи: а) датчик струму ACS712 [53]; б) модуль напруги ZMPT101B [54]; в) електромагнітне реле SRD-05VDC-SL-C [55]

Фундаментальним аспектом проєктування вимірювальних кіл для мереж 220 В є забезпечення безпеки цифрового ядра. Використання мікросхеми ACS712-20A базується на класичному електрофізичному явищі - ефекті Холла. При протіканні вимірюваного змінного струму I через вбудований мідний провідник виникає магнітне поле, вектор магнітної індукції B якого перпендикулярний до площини напівпровідникового кристала Холла. Відповідно до законів електродинаміки, на рухомі носії заряду у напівпровіднику починає діяти сила Лоренца:

$$F_L = q(v * B), \tag{2.2}$$

де q – заряд електрона;

v – швидкість дрейфу носіїв заряду.

Під дією цієї сили носії заряду відхиляються до країв кристала, створюючи поперечну різницю потенціалів (напругу Холла). Даний безконтактний метод забезпечує гальванічну ізоляцію між силовою та цифровою частинами системи.

Не менш важливим є фізичний механізм ізоляції виконавчого вузла. Керування електромагнітним реле відбувається через оптопару EL817. Передача сигналу в цьому компоненті здійснюється оптичним шляхом: логічний рівень від мікроконтролера активує інфрачервоний світлодіод (GaAs), який випромінює потік фотонів. Ці фотони долають прозорий діелектричний бар'єр і потрапляють на базу кремнієвого фототранзистора, викликаючи його відкриття (генерацію фотоструму).

Такий оптоелектронний бар'єр витримує напругу пробою понад 5000 В і повністю блокує проникнення високовольтних імпульсів (ЕРС самоіндукції), що виникають при розриві ланцюга живлення реле, у цифрову частину схеми.

Обґрунтування вибору модулів живлення вимагає розуміння фізики процесів перетворення напруги. Первинний перетворювач HLK-PM01 належить до класу імпульсних джерел живлення (SMPS - Switched-Mode Power Supply). На відміну від застарілих трансформаторних блоків, його робота базується на принципі широтно-імпульсної модуляції. Високовольтна змінна напруга (220 В) спочатку випрямляється діодним мостом і фільтрується високовольтним конденсатором. Далі високочастотний транзисторний ключ розсікає цю постійну напругу на імпульси з частотою в десятки кілогерц (зазвичай 30-100 кГц). Ці імпульси подаються на первинну обмотку мініатюрного імпульсного високочастотного трансформатора. Завдяки високій частоті, розміри осердя трансформатора вдалося зменшити в десятки разів при збереженні передаваної потужності (до 3 Вт). На вторинній обмотці напруга знову випрямляється та згладжується. Оптичний зворотний зв'язок всередині HLK-PM01 керує шпаруватістю (Duty Cycle) імпульсів ШІМ-контролера, забезпечуючи жорстку стабілізацію вихідної напруги на рівні 5 В незалежно від коливань напруги в первинній електромережі. Вторинне стабілізування за допомогою AMS1117-3.3

працює за іншим, лінійним принципом. Цей LDO-регулятор (Low Dropout) працює як динамічно змінний опір. Регулятор розсіює різницю потенціалів між вхідною та вихідною напругами 1.7 В (різниця між 5 В на вході та 3.3 В на виході) у вигляді теплової енергії. При споживанні системою струму у 345 мА, теплова потужність, що розсіюється на кристалі AMS1117, становить $P_{heat} = (5V - 3.3V) \cdot 0.345A = 0.58$ Вт. Це значення лежить у безпечних межах для корпусу SOT-223, проте вимагає наявності полігону охолодження на друкованій платі. Щодо оптопари EL817, яка використовується у модулі реле, критичним інженерним параметром є коефіцієнт передачі струму (CTR - Current Transfer Ratio). Він визначає відношення вихідного струму фототранзистора до вхідного струму інфрачервоного світлодіода. З часом, внаслідок деградації кристала світлодіода, CTR оптопари неминуче знижується. Саме тому при розрахунку струмообмежувального резистора (330 Ом) керуючого сигналу D23 було закладено підвищений струм (близько 10 мА), що гарантує надійне насичення транзистора 2N2222 і безвідмовне спрацювання реле навіть після кількох років безперервної експлуатації в умовах серверного приміщення.

2.2. Опис середовища програмування та моделювання

Розробка програмного забезпечення для вбудованих систем реального часу вимагає використання спеціалізованих інтегрованих середовищ розробки, які поєднують у собі редактор вихідного коду, систему автоматизації збирання проєкту, бібліотеки апаратної абстракції, крос-компілятори та засоби налагодження програмного забезпечення. Особливістю embedded-розробки є необхідність взаємодії програмного коду безпосередньо з апаратними ресурсами мікроконтролера: периферійними інтерфейсами, таймерами, контролерами переривань, модулями аналого-цифрового перетворення та мережевими підсистемами. У зв’язку з цим вибір програмного середовища безпосередньо

впливає на стабільність роботи системи, швидкість відлагодження та ефективність реалізації програмної логіки.

Розробка програмного забезпечення та моделювання роботи апаратної частини здійснюється у хмарній платформі Wokwi із використанням фреймворку Arduino Core for ESP32. Платформа поєднує редактор вихідного коду, систему компіляції, засоби віртуального моделювання електронних схем та інструменти налагодження програмного забезпечення. Такий підхід дозволяє реалізувати повний цикл розробки вбудованої системи - від написання програмного коду до тестування взаємодії периферійних модулів без необхідності постійного використання фізичного обладнання.

Конфігурація віртуальної схеми у Wokwi здійснюється за допомогою JSON-опису, у якому задаються типи електронних компонентів, параметри підключення та топологія електричних з’єднань між модулями системи. Такий підхід дозволяє швидко модифікувати структуру пристрою та автоматизувати процес тестування різних конфігурацій апаратної частини.

Основою програмної архітектури системи є фреймворк Arduino Core for ESP32, який виступає проміжним рівнем абстракції між прикладним програмним кодом та низькорівневими апаратними регістрами мікроконтролера. Даний фреймворк базується на офіційному SDK ESP-IDF (Espressif IoT Development Framework) та надає високорівневий API для взаємодії з цифровими інтерфейсами вводу/виводу, аналого-цифровими перетворювачами, апаратними таймерами, модулями PWM, стеком Wi-Fi/Bluetooth та файловою системою SPIFFS. Завдяки використанню готових бібліотек значно скорочується обсяг низькорівневого коду, необхідного для конфігурації периферії, що дозволяє зосередити основну увагу на реалізації алгоритмів вимірювання та моніторингу параметрів електромережі.

Незважаючи на те, що класична Arduino-модель програмування базується на використанні функцій setup() та loop(), внутрішня архітектура ESP32 автоматично створює окремі системні задачі для роботи мережевого стеку,

драйверів периферії та обслуговування бездротових інтерфейсів. Це дозволяє реалізовувати паралельне виконання декількох процесів одночасно: зчитування даних із сенсорів, обчислення середньоквадратичних значень струму та напруги, передачу телеметрії через Wi-Fi та контроль аварійних режимів роботи системи.

Для налагодження програмного забезпечення у Wokwi використовується вбудований емулятор послідовного інтерфейсу UART та віртуальний Serial Monitor, який дозволяє здійснювати моніторинг службових повідомлень, значень сенсорів і мережевих подій у режимі реального часу. Для забезпечення коректного цифрового відтворення синусоїдальної форми сигналів напруги та струму частота дискретизації аналого-цифрового перетворювача повинна щонайменше вдвічі перевищувати частоту вимірюваного сигналу відповідно до теореми Котельникова–Шеннона. У розробленій системі частота дискретизації встановлюється в діапазоні 1–5 кГц, що забезпечує достатню точність обчислення середньоквадратичних значень та аналізу параметрів електромережі. Для зменшення впливу високочастотних перешкод та ефекту накладання спектрів (aliasing) перед входами АЦП можуть використовуватися RC-фільтри нижніх частот.

Через UART0 виконується передача службових повідомлень, діагностичних даних, результатів вимірювань та інформації про стан мережевого підключення у режимі реального часу. Використання послідовного монітора суттєво спрощує процес локалізації помилок, дозволяє аналізувати часові затримки виконання окремих задач та контролювати коректність функціонування алгоритмів цифрової обробки сигналів

Окремою перевагою платформи ESP32 є підтримка механізму OTA (Over-The-Air) оновлення прошивки. Даний механізм дозволяє виконувати дистанційне оновлення програмного забезпечення через бездротову мережу Wi-Fi без необхідності фізичного підключення пристрою через USB-інтерфейс. Використання OTA є особливо важливим для систем моніторингу, які встановлюються у важкодоступних місцях або працюють у безперервному

режимі, оскільки дозволяє оперативно оновлювати функціонал системи та усувати програмні помилки без демонтажу обладнання.

Для передачі телеметричних даних та інтеграції системи у мережеву інфраструктуру використовуються сучасні протоколи обміну даними MQTT та HTTP. Протокол MQTT (Message Queuing Telemetry Transport) є легковаговим мережевим протоколом, оптимізованим для систем Інтернету речей та передачі невеликих пакетів телеметрії з мінімальними накладними витратами. Його використання дозволяє реалізувати архітектуру publish/subscribe із централізованим MQTT-брокером, через який здійснюється обмін даними між мікроконтролером та серверною частиною системи моніторингу. У свою чергу, HTTP-протокол може застосовуватися для передачі даних до веб-сервісів, REST API або локальних інформаційних панелей диспетчеризації.

Враховуючи підвищені ризики при роботі з високовольтною мережею змінного струму 220 В, процес початкового конструювання, налагодження програмної логіки та тестування аварійних режимів виконується у спеціалізованому середовищі віртуального симулювання Wokwi. Wokwi є сучасною хмарною платформою для моделювання схем Інтернету речей та вбудованих цифрових систем, яка підтримує широкий спектр мікроконтролерів, периферійних модулів та електронних компонентів.

На відміну від класичних SPICE-симуляторів, орієнтованих переважно на аналіз аналогових електричних схем, архітектура Wokwi базується на повному програмному емулюванні системи команд цільового процесора. У випадку ESP32 емулятор виконує покрокову інтерпретацію машинних інструкцій архітектури Xtensa LX6, моделюючи роботу процесорних ядер, регістрів загального призначення, кеш-пам’яті, контролерів переривань та підсистеми управління пам’яттю MMU (Memory Management Unit). Завдяки цьому поведінка симульованого мікроконтролера максимально наближена до роботи фізичного пристрою.

Ключовою перевагою Wokwi є можливість виконання нативно скомпільованого бінарного файлу прошивки (.bin), сформованого компілятором GCC для архітектури Xtensa. Це дозволяє тестувати реальний програмний код без спрощення логіки чи переписування окремих модулів спеціально для середовища симуляції. У процесі моделювання емулятор підтримує роботу апаратних таймерів, GPIO, шин SPI, I2C, UART, аналого-цифрових перетворювачів, PWM-модулів та системи переривань, що дозволяє перевіряти коректність взаємодії програмного забезпечення з периферією.

Унікальною функцією платформи є наявність віртуального мережевого шлюзу Virtual Wi-Fi Gateway. Симульований мікроконтролер ESP32 може підключатися до віртуальної точки доступу Wokwi-GUEST та отримувати повноцінний доступ до глобальної мережі Інтернет. Завдяки цьому можливо тестувати роботу мережевих протоколів, MQTT-з'єднань, HTTP-запитів та взаємодію із серверною частиною системи без використання фізичного Wi-Fi роутера чи реального мережевого обладнання.

Використання віртуального моделювання суттєво підвищує безпечність розробки системи моніторингу електромережі. Тестування алгоритмів аварійного вимкнення навантаження, контролю перевантажень та обробки помилок може здійснюватися без фізичного контакту із високовольтними елементами мережі 220 В, що мінімізує ризик пошкодження апаратури або виникнення небезпечних аварійних ситуацій під час налагодження.

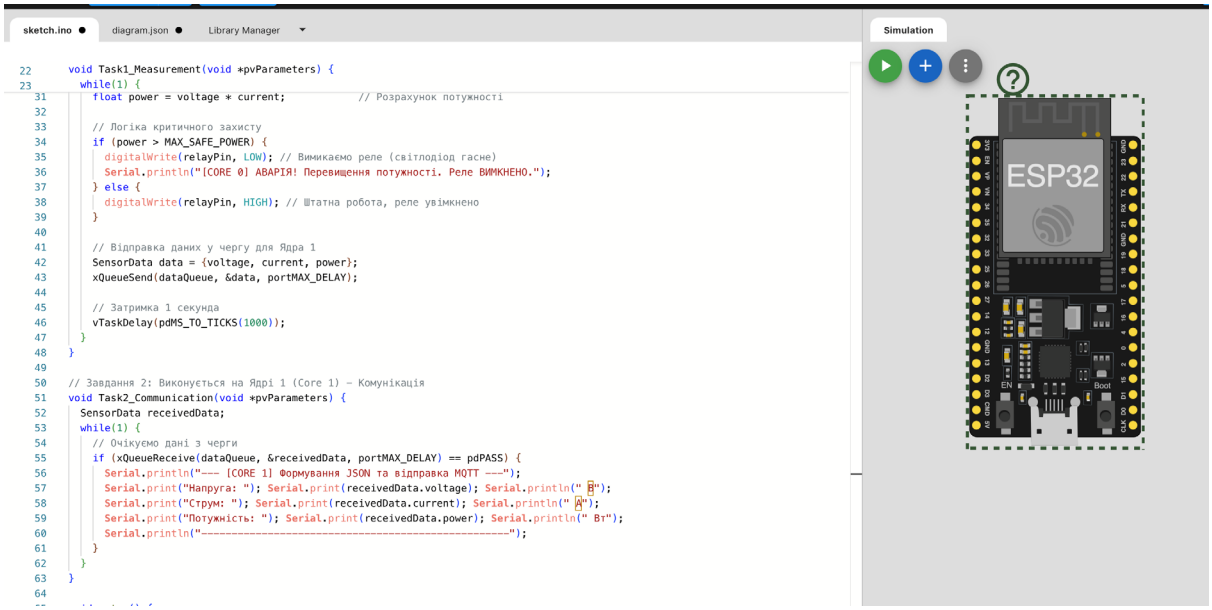


Рисунок 2.3 – Інтерфейс середовища розробки та віртуального моделювання Wokwi

Оскільки розробка програмного забезпечення здійснюється на персональному комп’ютері з архітектурою x86 або x64, а цільовим апаратним середовищем є мікроконтролер ESP32 з архітектурою Xtensa LX6, процес створення виконуваного програмного коду базується на механізмі крос-компіляції (Cross-compilation). Крос-компілятор дозволяє генерувати машинний код для архітектури, відмінної від архітектури комп’ютера, на якому виконується розробка.

Першим етапом формування виконуваного файлу є препроцесинг (Preprocessing). На цьому етапі спеціальна утиліта обробляє директиви препроцесора, що починаються символом #, зокрема #include, #define, #ifdef та інші. Відбувається підключення заголовних файлів зовнішніх бібліотек, розгортання макросів, умовна компіляція та формування єдиного текстового представлення програмного коду.

Наступним етапом є безпосередньо компіляція (Compilation), під час якої компілятор GCC (GNU Compiler Collection) виконує синтаксичний та семантичний аналіз програмного коду мовою C++. На основі отриманого синтаксичного дерева формується проміжне представлення програми, після чого

генерується асемблерний код, специфічний для системи команд архітектури Xtensa LX6. На цьому ж етапі застосовуються алгоритми оптимізації коду, спрямовані на зменшення розміру прошивки, оптимізацію використання оперативної пам’яті та підвищення швидкодії виконання інструкцій. Найчастіше використовуються оптимізаційні прапорці -O2 або -Os, які забезпечують компроміс між продуктивністю та компактністю програмного коду.

Третім етапом є асемблювання (Assembly), під час якого мнемоніки асемблерного коду перетворюються у машинні інструкції процесора та формуються об’єктні файли формату .o. Кожен такий файл містить окремі секції коду, констант та службової інформації, необхідної для подальшого компонування програми.

Завершальним етапом процесу крос-компіляції є лінкування (Linking). На цьому етапі лінкувальник об’єднує всі об’єктні файли проєкту із попередньо скомпільованими бібліотеками, драйверами периферійних інтерфейсів та мережевими модулями. Відповідно до сценарію компонування (Linker Script) виконується розподіл програмного коду, глобальних змінних, констант і службових структур між фізичними областями SRAM, ROM та SPI Flash-пам’яті мікроконтролера ESP32. Результатом роботи лінкувальника є формування єдиного виконуваного бінарного файлу прошивки формату .bin, готового до завантаження у флеш-пам’ять мікроконтролера через UART або OTA-оновлення.

2.3. Опис мови системного програмування

Основним інструментом створення керуючого програмного забезпечення (Firmware) для пристрою моніторингу обрано мову системного програмування C++. Вибір C/C++ є стандартом де-факто у галузі Embedded Systems, що зумовлено високою швидкодією зкомпільованого коду, детермінованістю

виконання операцій та можливістю прямого доступу до апаратної пам'яті за допомогою вказівників.

На відміну від програмування для настільних ПК, використання C++ для мікроконтролерів має специфічні обмеження, пов'язані з дефіцитом оперативної пам'яті (SRAM). У розробленому коді активно застосовуються принципи об'єктно-орієнтованого програмування (ООП), зокрема інкапсуляція. Стан датчиків та їхні методи обробки упаковуються у логічні структури (struct або class), що підвищує читабельність та модульність коду.

Однак, з метою уникнення фрагментації пам'яті (Memory Fragmentation) та витоків пам'яті (Memory Leaks), у системному програмуванні на C++ суворо обмежується або повністю забороняється використання динамічного виділення пам'яті (оператори new та delete, функції malloc() та free()) на етапі основного робочого циклу. Замість цього застосовується статичне виділення пам'яті на етапі ініціалізації пристрою. Крім того, не використовуються механізми обробки виключень (try-catch), оскільки вони суттєво збільшують розмір скомпільованого бінарного файлу та знижують швидкість виконання.

Для форматування даних перед передачею по мережі у C++ використовується бібліотека серіалізації. У даному проекті дані структуруются у текстовий формат JSON , що забезпечує легку інтеграцію пристрою з веб-сервісами та базами даних [59].

Мова C++ забезпечує ефективну взаємодію програмного забезпечення з апаратними ресурсами мікроконтролера ESP32. У системах embedded-програмування програмний код безпосередньо працює з GPIO, аналого-цифровими перетворювачами, UART, SPI, I2C та мережевими модулями, тому важливими є швидкодія виконання та контроль використання пам’яті.

На відміну від класичних програм для персональних комп’ютерів, firmware для вбудованих систем повинно працювати в умовах обмежених апаратних ресурсів та забезпечувати стабільну реакцію на зовнішні події у режимі реального часу. Саме тому під час системного програмування особлива увага

приділяється оптимізації програмного коду, ефективному використанню оперативної пам’яті та мінімізації затримок обробки даних.

В основі системного програмування C++ для вбудованих систем лежить суворий контроль за використанням оперативної пам’яті SRAM. Оскільки її обсяг у мікроконтролерах є обмеженим, під час розробки firmware зазвичай застосовується статичне виділення пам’яті та мінімізується використання надмірних динамічних структур. Недостатній контроль використання пам’яті може призвести до нестабільної роботи системи або аварійного завершення виконання програми.

Під час роботи з апаратними регістрами та периферійними модулями активно використовуються вказівники та спеціалізовані бібліотеки Arduino Core for ESP32. Крім цього, при роботі зі змінними, значення яких можуть змінюватися апаратними модулями незалежно від основного потоку виконання програми, використовується ключове слово volatile. Воно забороняє компілятору кешувати значення змінної у внутрішніх регістрах процесора та забезпечує зчитування актуальних даних безпосередньо з оперативної пам’яті.

Критично важливим параметром embedded-систем є швидкість реакції на зовнішні події та апаратні сигнали. Архітектура ESP32 дозволяє забезпечувати мінімальні затримки під час обробки вимірювальних даних, що є важливим для систем моніторингу електромереж та аварійного захисту.

Для формування структурованих телеметричних даних у подальшому може використовуватися формат JSON, який забезпечує зручну інтеграцію системи з MQTT-брокерами, веб-сервісами та серверними інформаційними системами.

Процес створення виконуваного файлу прошивки включає:

1) препроцесинг;
2) компіляцію;
3) асемблювання;
4) лінкування.

На етапі лінкування виконується об’єднання всіх об’єктних файлів проєкту із системними бібліотеками та драйверами периферійних інтерфейсів. Відповідно до сценарію компонування здійснюється розподіл програмного коду, глобальних змінних та констант між фізичними областями SRAM, ROM та SPI Flash-пам’яті мікроконтролера ESP32.

Результатом компіляції є формування бінарного файлу прошивки формату .bin, який завантажується до флеш-пам’яті ESP32 через UART або OTA-оновлення.

2.4 Висновки до другого розділу

У другому розділі було розглянуто апаратну та програмну базу розроблюваної системи моніторингу енергоспоживання на основі мікроконтролера ESP32. Проведено аналіз характеристик основних компонентів системи та обґрунтовано доцільність їх використання у межах поставленої задачі.

У результаті порівняння мікроконтролерних платформ було підтверджено вибір ESP32 як центрального обчислювального вузла системи. Визначено, що архітектурні особливості мікроконтролера, зокрема висока тактова частота, значний обсяг оперативної пам’яті, інтегровані модулі бездротового зв’язку та багатоканальний аналого-цифровий перетворювач, дозволяють реалізувати одночасний збір, обробку та передачу телеметричних даних у режимі реального часу.

Також було досліджено принцип роботи та технічні характеристики сенсорів ACS712 і ZMPT101B, які використовуються для вимірювання параметрів електромережі. Визначено, що використання гальванічної розв’язки забезпечує необхідний рівень електробезпеки та захисту низьковольтної частини системи від впливу високої напруги.

Окрему увагу приділено системі живлення, модулю реле та принципам взаємодії периферійних компонентів із мікроконтролером. Було встановлено, що використання імпульсного AC/DC-перетворювача HLK-PM01 та стабілізатора AMS1117 дозволяє забезпечити стабільне живлення всієї системи.

Крім апаратної складової, у розділі розглянуто середовище розробки та моделювання Wokwi. Визначено, що дане середовище дозволяє виконувати симуляцію роботи ESP32, тестування алгоритмів обробки даних, налагодження взаємодії між компонентами та перевірку логіки аварійного реагування без використання фізичного обладнання.

Таким чином, у другому розділі було сформовано повну технічну основу для подальшої реалізації програмно-апаратного комплексу моніторингу енергоспоживання.

3 КОНСТРУЮВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ МОНІТОРИНГУ

3.1 Архітектура пристрою

Процес конструювання починається з розробки загальної архітектури. Архітектура визначає взаємодію апаратної та програмної складових. Запропонована структура системи моніторингу складається з п’яти логічних рівнів.

Перший рівень є сенсорним. Він відповідає за зняття первинних показників з мережі змінного струму. Тут розташовані фізичні датчики напруги та струму. Вони перетворюють високовольтні параметри мережі у слабкострумові аналогові сигнали.

Другий рівень є обчислювальним. Його основу складає мікроконтролер ESP32. Він отримує аналогові сигнали та перетворює їх у цифровий формат. Процесор виконує всі математичні розрахунки та керує логікою роботи всього пристрою.

Третій рівень є виконавчим. Цей блок містить електромеханічне реле та транзисторний ключ керування. Він відповідає за фізичне розірвання ланцюга живлення при виникненні аварійної ситуації. Мікроконтролер керує цим рівнем за допомогою цифрових сигналів.

Четвертий рівень забезпечує комунікацію. Він відповідає за підтримку протоколів бездротової передачі даних. Цей рівень реалізується завдяки вбудованому радіомодулю мікроконтролера. Пристрій підключається до локальної мережі та встановлює зв’язок з віддаленим сервером.

П’ятий рівень є серверним. Він приймає готові телеметричні дані для їх подальшого збереження та візуалізації. Взаємодія між цими п’ятьма рівнями відбувається безперервно у режимі реального часу. Інформація проходить шлях від фізичного вимірювання до хмарної бази даних за частки секунди.

3.2 Розробка структурної та принципової схеми програмно-технічного засобу

Структурна схема програмно-технічного засобу моніторингу електромережі відображає взаємозв'язки між основними підсистемами системи та її функціональну організацію. Відповідно до прийнятої архітектурної концепції (розділ 3.1), система розподілена на два рівні: локальний та мережевий.

Локальний рівень об'єднує апаратні компоненти, безпосередньо залучені до первинного вимірювання та управління. Центральним елементом є мікроконтролер ESP32(DevKit C V4), який виконує функції обчислювального ядра та координує роботу всієї периферії. До нього підключено такі функціональні вузли:

1) перетворювач мережевого живлення HLK-PM01, який здійснює первинне перетворення напруги змінного струму 220 В у стабілізовану постійну напругу 5В. Він є єдиним вузлом, що гальванічно з'єднаний з мережею змінного струму, та живить усі інші компоненти схеми;

2) лінійний стабілізатор AMS1117-3.3, який формує вторинну напругу живлення 3.3В безпосередньо для логічної частини мікроконтролера ESP32, забезпечуючи додаткову фільтрацію пульсацій та захист від перевищення рівня напруги;

3) модуль напруги на базі трансформаторного перетворювача ZMPT101B, який виконує гальванічно ізольоване масштабування змінної мережевої напруги до рівня, придатого для оцифрування АЦП (0-3,3 В). Вихідний сигнал надходить на аналоговий вхід GPIO35 мікроконтролера;

4) датчик струму ACS712-20A, що реалізує безконтактне вимірювання струму навантаження на основі ефекту Холла з гальванічною ізоляцією до 2,1 кВ. Аналоговий вихідний сигнал датчика через прецизійний резистивний дільник подається на аналоговий вхід GPIO34;

5) електромагнітне реле SRD-05VDC-SL-C, яке є виконавчим елементом схеми та забезпечує комутацію навантаження (ланцюга 220 В). Керування реле здійснюється мікроконтролером через цифровий вихід GPIO23 з гальванічною розв’язкою.

Мережевий рівень забезпечує взаємодію пристрою із зовнішнім середовищем. Мікроконтролер ESP32 через внутрішній модуль Wi-Fi (802.11 b/g/n) встановлює бездротове з’єднання з мережею за протоколом DHCP. У симуляційному середовищі Wokwi (Vitual Wi-Fi Gateway), що надає симульованому мікроконтролеру двосторонній доступ до глобальної мережі Інтернет без фізичного маршрутизатора. Результати вимірювань та розрахунків передаються у вигляді структурованих JSON-пакетів і виводяться до системної консолі – термінала симулятора.

Структурна схема, наведена на рис. 3.1.

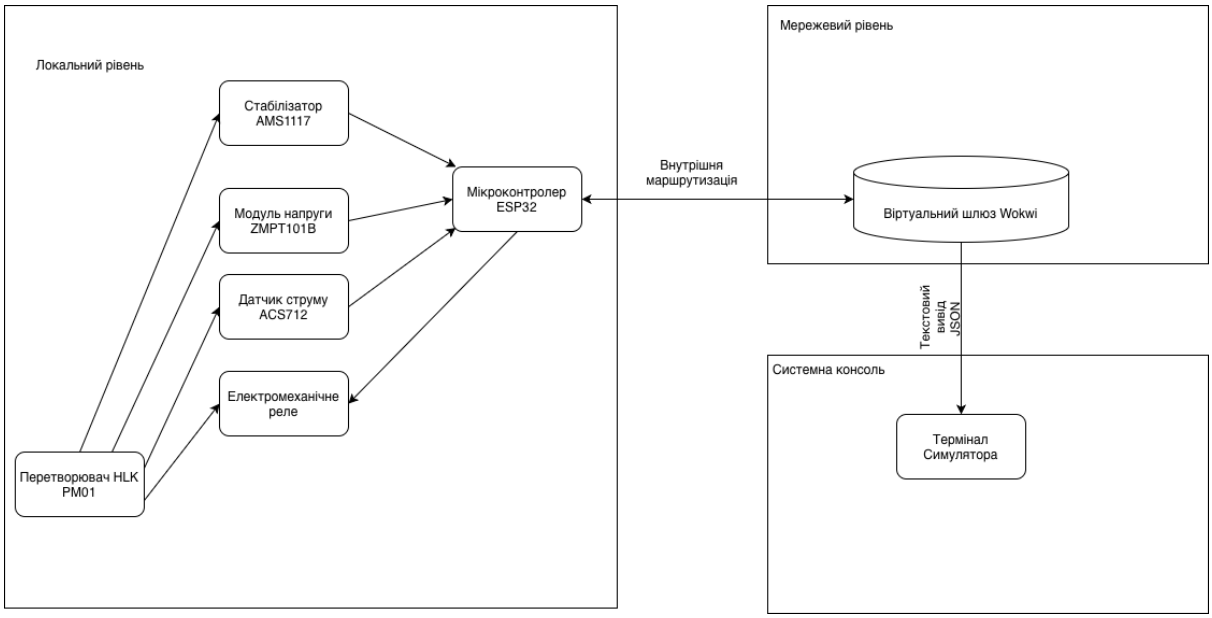


Рисунок 3.1 – Структурна схема програмно-технічного засобу моніторингу електромережі

Електрична принципова схема розроблена на основі обраної елементної бази та вимог, сформульованих у розділах 2 та 3.1. Схема деталізує електричні з’єднання між усіма компонентами пристрою, типи та номінали пасивних

елементів, розводку ліній живлення та сигнальних кіл. Схема виконана у середовищі EasyEDA та наведена на рис 3.2.

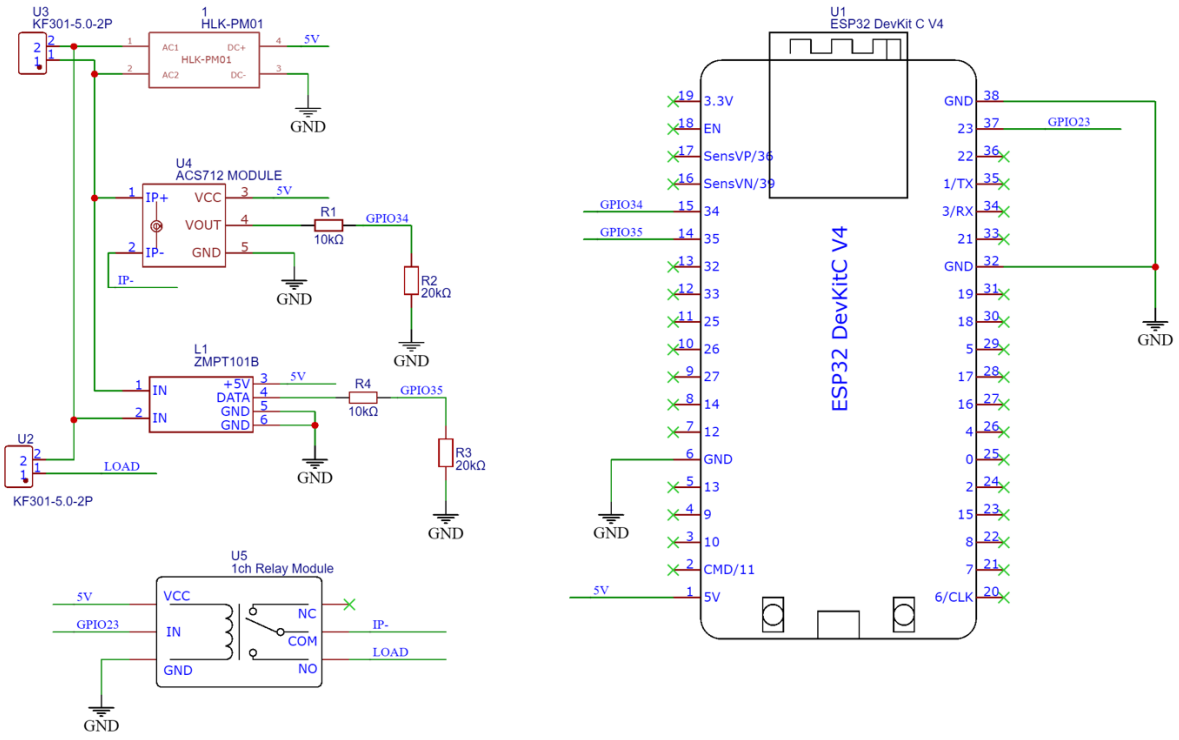


Рисунок 3.2 – Електрична принципова схема пристрою моніторингу

Розглянемо детально кожний функціональний вузол принципової схеми.

Первинне живлення подається від мережі 220 В через роз'єм KF301-5.0-2P(U3) до модуля AC/DC-перетворювача HLK-PM01. Виводи AC1 та AC2 підключені до фазового та нульового провідників мережі відповідно. На виходах DC+ та DC- модуля формується стабілізована напруга +5 В, яка надходить на силову шину живлення 5 В усієї схеми.

Вторинна напруга 3,3 В для живлення логічної частини мікроконтролера ESP32 формується лінійним LDO-регулятором AMS1117-3.3. Вхід регулятора підключений до шини +5 В, вихід – до шини +3,3 В. Згідно з рекомендаціями виробника, для забезпечення стабільності роботи регулятора паралельно входу та виходу встановлюються розв'язувальні конденсатори.

Модуль датчика струму ACS712-20A (U4) підключений послідовно в розрив навантажувального ланцюга. Вивід VCC датчика живиться від шини +5 В, вивід GND - до спільного проводу. Вимірювальний струм протікає між виводами IP+ та IP-, які з'єднані із відповідними затискачами. Аналоговий вихід VOUT датчика, де формується напруга у діапазоні 0,5-4,5 В пропорційно до вимірюваного струму, не може бути безпосередньо підключений до аналогового входу ESP32 через перевищення максимально допустимого рівня 3,3 В.

Для узгодження рівнів між виходом ACS712 та аналоговим входом GPIO34 мікроконтролера застосований прецизійний резистивний дільник напруги, що складається з резисторів R1 (10 кОм) та R2 (20 кОм). Коефіцієнт ділення дільника розраховується за формулою:

$$K = \frac{R_2}{R_1 + R_2} = \frac{20}{10 + 20} = 0{,}667, \qquad (3.1)$$

Таким чином, максимальне вхідне значення 4,5 В масштабується до рівня $4{,}5 \times 0{,}667 \approx 3{,}0$ В, що знаходиться у межах безпечного та лінійного діапазону роботи АЦП мікроконтролера. Вивід GPIO34 є вхідним (input-only) піном ESP32, що додатково запобігає випадковій подачі напруги з боку мікроконтролера.

Модуль трансформаторного перетворювача напруги ZMPT101B (L1) підключений паралельно до навантажувального ланцюга (між виводами роз'єму U2 типу KF301-5.0-2P). Виводи IN1 та IN2 модуля підключені до мережевої напруги, а виводи +5V та GND - до шин живлення. Вихідний аналоговий сигнал знімається з виводу DATA.

Аналогічно до вузла вимірювання струму, для захисту аналогового входу GPIO35 мікроконтролера застосовується резистивний дільник напруги на резисторах R4 (10 кОм) та R3 (20 кОм) з ідентичним коефіцієнтом ділення 0,667. Це гарантує, що вихідний сигнал ZMPT101B не перевищить допустимого рівня 3,3 В на вході АЦП мікроконтролера.

Комутація навантаження здійснюється за допомогою одноканального модуля реле (U5, 1ch Relay Module), до складу якого входять електромеханічне реле SRD-05VDC-SL-C, транзисторний ключ та захисна оптопара. Вивід VCC модуля підключений до шини +5 В, GND - до спільного проводу. Керуючий логічний сигнал надходить з цифрового виходу GPIO23 мікроконтролера на вхід IN модуля.

Нормально розімкнений контакт (NO) реле включений послідовно в розрив ланцюга IP– (повернення струму від навантаження до датчика ACS712). Загальний контакт (COM) підключений до виходу навантаження (лінія LOAD). Такий спосіб підключення реалізує можливість програмного відключення навантаження від мережі за командою мікроконтролера - наприклад, у разі виявлення аварійного перевищення струму або напруги.

Нормально замкнений контакт (NC) залишається не підключеним (показаний як розрив на схемі), оскільки у даній реалізації функція примусового увімкнення навантаження при знеструмленні керуючого ланцюга не передбачена.

Центральним елементом принципової схеми є мікроконтролер ESP32 DevKitC V4 (U1). Живлення мікроконтролерного модуля здійснюється через вивід 5V (пін 1), підключений до шини +5 В. Внутрішній регулятор модуля формує з цієї напруги рівень 3,3 В для живлення мікросхеми ESP32 та виводить його на пін 3.3V (пін 19) для живлення периферії при потребі.

Лінії GND (піни 6, 32, 38) підключені до спільного нульового проводу схеми. Аналогові входи GPIO34 (пін 15) та GPIO35 (пін 14) використовуються для приймання сигналів від датчиків струму та напруги відповідно. Цифровий вихід GPIO23 (пін 37) формує керуючий сигнал HIGH/LOW для вмикання та вимикання реле. Решта виводів мікроконтролера залишаються незадіяними в даній реалізації та позначені на схемі символом «×».

Сукупність описаних вузлів утворює замкнений функціональний ланцюг: мережева напруга вимірюється та комутується на апаратному рівні, первинні

аналогові сигнали обробляються АЦП мікроконтролера, а розраховані метричні показники передаються через бездротовий інтерфейс Wi-Fi на верхній рівень системи моніторингу.

3.3 Розробка алгоритмів функціонування програмно-технічного засобу

Наступним етапом розробки стало формування алгоритмічної логіки роботи програмно-технічного засобу. Якщо у підрозділі 3.2 було показано, яким чином компоненти пристрою фізично та функціонально пов’язані між собою, то в даному підрозділі розглянуто, у якій послідовності ці компоненти взаємодіють під керуванням програмного забезпечення.

Алгоритми розроблялися відповідно до створеної моделі у середовищі Wokwi та до наявної програмної реалізації для мікроконтролера ESP32 мовою C++. У поточному прототипі основна увага зосереджена на трьох взаємопов’язаних процесах:

1) початковій ініціалізації апаратних і програмних ресурсів;

2) циклічному зчитуванні параметрів, обчисленні потужності та реалізації аварійного захисту;

3) передаванні отриманих вимірювальних даних між ядрами ESP32 та їх подальшому виведенні до системної консолі симулятора.

Саме така послідовність відповідає реалізованій багатозадачній структурі системи: Core 0 виконує вимірювально-захисну функцію, а Core 1 приймає підготовлені дані з черги FreeRTOS і забезпечує їх подальше представлення у зручній формі для налагодження. Такий поділ уже був закладений у архітектуру пристрою, описану в підрозділах 3.1–3.2, де окремо виділено вимірювальний модуль, виконавчий модуль керування реле та механізм міжпроцесорного обміну через структуру SensorData і чергу dataQueue.

3.3.1. Алгоритм ініціалізації пристрою

Початковий етап роботи системи полягає у підготовці мікроконтролера ESP32 до подальшого виконання двох паралельних програмних задач. На відміну від простих однопотокових мікроконтролерних програм, у яких основна логіка зосереджується у функції loop(), у розробленому пристрої після запуску відразу створюється багатозадачна структура на базі FreeRTOS.

Алгоритм ініціалізації пристрою наведено на рис. 3.3.

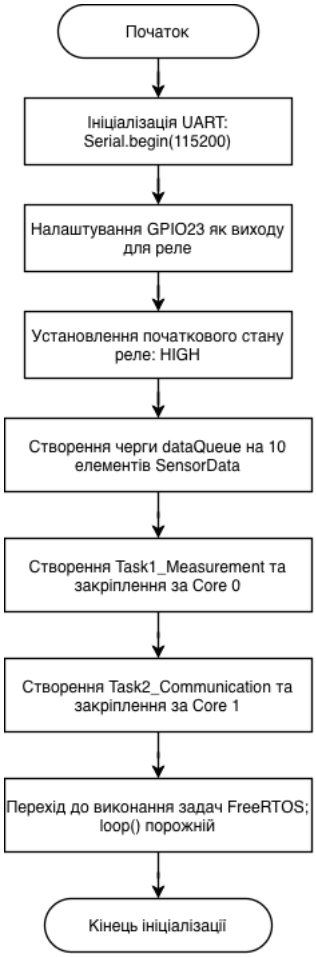


Рисунок 3.3 – Алгоритм ініціалізації пристрою

Алгоритм починається з подачі живлення на пристрій та запуску прошивки. Після цього виконується ініціалізація послідовного інтерфейсу за допомогою команди Serial.begin(115200).

Використання UART у даній реалізації має практичне значення, оскільки через Serial Monitor середовища Wokwi відображаються результати вимірювань, службові повідомлення про роботу комунікаційного завдання, а також сповіщення про аварійне перевищення потужності. Це дає змогу безпосередньо в середовищі моделювання контролювати поточні значення параметрів і момент спрацювання захисної логіки.

Наступним кроком виконується налаштування цифрового виводу GPIO23, який у розробленій схемі використовується для керування виконавчим модулем. У фізичній реалізації цей сигнал призначений для керування реле, а у віртуальній моделі Wokwi його стан відображається за допомогою світлодіода зі струмообмежувальним резистором. Таким чином, світлодіод візуально демонструє, чи перебуває система у штатному режимі, чи спрацював захисний алгоритм. Відповідність GPIO23 виконавчому каналу була закладена ще під час створення принципової схеми пристрою.

У програмі ця частина ініціалізації реалізована такими командами:

```
pinMode(relayPin, OUTPUT);
digitalWrite(relayPin, HIGH).
```

Перша команда переводить GPIO23 у режим цифрового виходу, а друга задає початковий логічний стан HIGH. У межах створеної моделі це відповідає нормальному режиму роботи, за якого реле умовно вважається увімкненим, а світлодіод - активним. Така логіка є обґрунтованою: одразу після запуску пристрій не блокує живлення навантаження, а лише переходить до спостереження за показниками та вмикає захист у разі перевищення встановленого порога.

Після налаштування виконавчого виходу створюється черга FreeRTOS:

```
dataQueue = xQueueCreate(10, sizeof(SensorData)).
```

Черга dataQueue використовується як буфер обміну між двома незалежними задачами. Її глибина становить десять елементів, а кожний елемент має тип SensorData. Відповідна структура містить три поля:

```
struct SensorData {
```

```
  float voltage;
  float current;
  float power;
};
```

Отже, після кожного циклу вимірювання перше ядро передає на друге не окремі значення, а повноцінний набір взаємопов'язаних параметрів: напругу, струм і розраховану потужність. Такий підхід спрощує подальшу обробку інформації та гарантує, що параметри надходять до другої задачі як єдиний логічний пакет. У підрозділі 3.2 цей механізм уже було визначено як основу міжпроцесорного обміну в межах багатоядерної програмної архітектури.

Завершальним етапом ініціалізації є створення двох паралельних завдань:

```
xTaskCreatePinnedToCore(Task1_Measurement, "Task1", 4096, NULL, 1, NULL, 0);
xTaskCreatePinnedToCore(Task2_Communication, "Task2", 4096, NULL, 1, NULL, 1).
```

Перше завдання Task1_Measurement закріплюється за Core 0 та відповідає за зчитування аналогових сигналів, розрахунок потужності, перевірку аварійної умови й керування виходом GPIO23. Друге завдання Task2_Communication закріплюється за Core 1 і виконує приймання вже сформованих даних із черги з подальшим їх виведенням до системної консолі.

Для кожної задачі задається стек обсягом 4096 байтів та однаковий пріоритет. У межах розробленого прототипу цього достатньо, оскільки обидві задачі виконують чітко визначені операції без складних вкладених обчислень або значних буферів локальних даних.

Після створення задач функція loop() залишається порожньою:

```
void loop() {
}.
```

Це означає, що уся прикладна логіка пристрою реалізується не через стандартний цикл Arduino, а через окремі задачі FreeRTOS. Така організація відповідає загальній концепції пристрою, відповідно до якої критичні вимірювальні операції та виведення телеметрії не змішуються в одному

послідовному блоці, а виконуються паралельно на різних ядрах ESP32. Обґрунтування доцільності такого підходу вже було наведено у другому розділі під час опису апаратних і програмних можливостей ESP32.

3.3.2. Алгоритм вимірювання параметрів, розрахунку потужності та аварійного захисту

Основна прикладна логіка розробленого програмно-технічного засобу реалізована в задачі Task1_Measurement, яка виконується на Core 0. Саме цей алгоритм безпосередньо реалізує ключове призначення пристрою - спостереження за умовним рівнем енергоспоживання контрольованого вузла та автоматичне реагування у випадку перевищення встановленого допустимого значення потужності.

Алгоритм вимірювання параметрів, розрахунку потужності та аварійного захисту наведено на рис. 3.4.

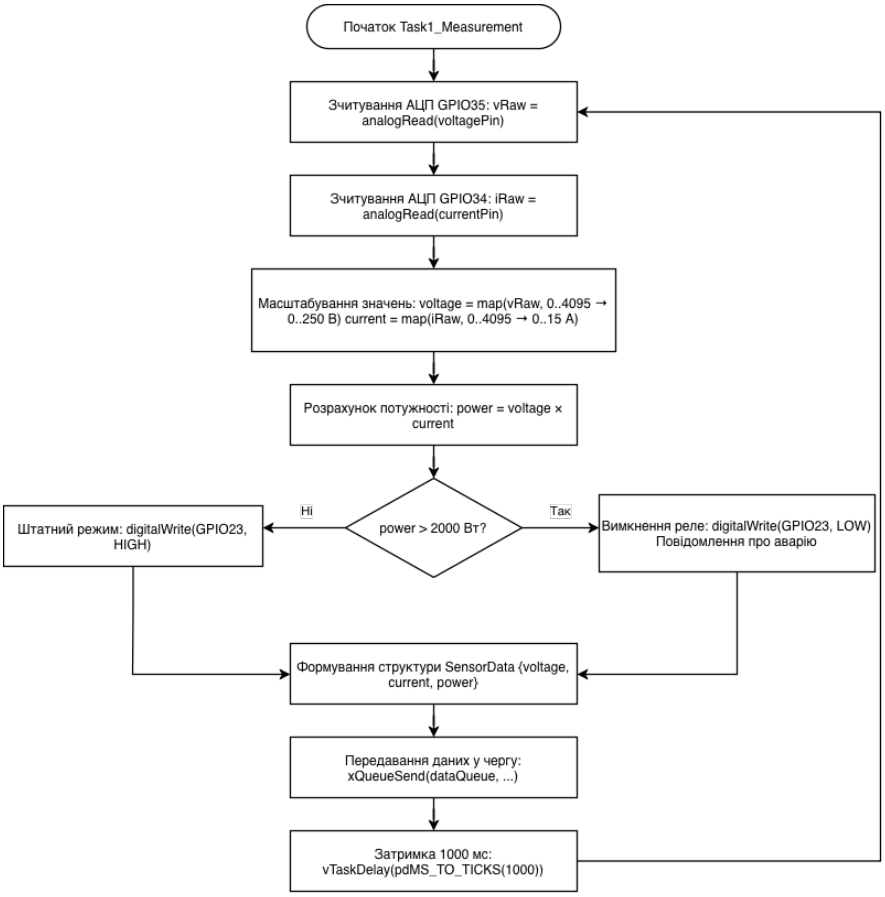


Рисунок 3.4 – Алгоритм вимірювання параметрів, розрахунку потужності та аварійного захисту

Робота задачі організована у вигляді нескінченного циклу:

```
while(1) {
    ...
}.
```

Це означає, що після завершення одного циклу вимірювання алгоритм автоматично повертається до початку та повторює всі дії знову. Завдяки цьому пристрій постійно оновлює інформацію про поточний стан модельованого навантаження.

На першому етапі циклу виконується зчитування двох аналогових каналів:

```
int vRaw = analogRead(voltagePin);
int iRaw = analogRead(currentPin).
```

У програмі встановлено такі відповідності між вимірювальними величинами та контактами ESP32:

```
const int currentPin = 34;
const int voltagePin = 35.
```

Отже, значення, що моделює струм, надходить із GPIO34, а значення, що моделює напругу, - із GPIO35. Цей вибір узгоджується з розробленою електричною принциповою схемою, де лінія струму пов'язана з каналом GPIO34, а лінія напруги - з GPIO35. У фізичному варіанті пристрою ці канали призначені для сигналів від датчика струму ACS712 та модуля контролю напруги ZMPT101B. У середовищі Wokwi їхню поведінку відтворюють два потенціометри, що дає змогу вручну імітувати різні значення аналогових сигналів та перевіряти реакцію алгоритму.

Функція analogRead() повертає цифрове значення в діапазоні від 0 до 4095. Такий діапазон відповідає 12-бітному аналого-цифровому перетворювачу ESP32. Проте для перевірки роботи системи важливо працювати не лише з сирими значеннями АЦП, а й з величинами, що мають прикладний зміст. Саме тому наступним кроком виконується їх програмне масштабування:

```
float voltage = map(vRaw, 0, 4095, 0, 250);
float current = map(iRaw, 0, 4095, 0, 15).
```

У межах створеної моделі весь діапазон зчитаного сигналу напруги перераховується в умовний інтервал 0–250 В, а сигнал струму - в інтервал 0–15 А. Важливо зазначити, що на даному етапі йдеться саме про модельне масштабування, яке використовується для перевірки логіки програми в симуляторі. Воно не є процедурою точного калібрування реальних сенсорів, оскільки у Wokwi роль вимірювальних модулів виконують потенціометри. Такий підхід цілком відповідає практичній меті поточного етапу -перевірити програмну реакцію системи на зміну вхідних аналогових сигналів.

Після формування значень напруги та струму обчислюється потужність:

```
float power = voltage * current.
```

У програмній моделі використано базове співвідношення:

$$P = U \cdot I, \tag{3.2}$$

де P – потужність, Вт;

U – значення напруги, В;

I – значення струму, А.

Цей розрахунок є центральним для подальшої логіки захисту, оскільки саме величина потужності використовується як критерій оцінювання режиму роботи системи. У підрозділі 3.1 уже було визначено, що пристрій повинен не лише окремо реєструвати струм і напругу, а й формувати на їх основі показник поточного споживання.

Після розрахунку потужності система переходить до перевірки аварійної умови. У коді задано граничне значення:

```
const float MAX_SAFE_POWER = 2000.0.
```

Таким чином, потужність 2000 Вт використовується як контрольний поріг. У кожному циклі алгоритм порівнює поточне значення power із цим лімітом:

```
if (power > MAX_SAFE_POWER) {
    digitalWrite(relayPin, LOW);
    Serial.println("[CORE 0] АВАРІЯ! Перевищення потужності. Реле ВИМКНЕНО.");
} else {
    digitalWrite(relayPin, HIGH);
}.
```

Якщо потужність перевищує встановлений поріг, на GPIO23 подається низький логічний рівень LOW. У моделі Wokwi це призводить до вимкнення світлодіода, який візуально імітує спрацювання релейного каналу. Одночасно в Serial Monitor формується повідомлення:

```
[CORE 0] АВАРІЯ! Перевищення потужності. Реле ВИМКНЕНО.
```

Це дозволяє не лише побачити зміну стану виконавчого елемента на схемі, а й підтвердити причину спрацювання захисту на програмному рівні.

Якщо значення потужності не перевищує 2000 Вт, виконується інша гілка алгоритму:

```
digitalWrite(relayPin, HIGH).
```

У цьому випадку вихід GPIO23 переводиться у високий логічний стан, тобто система зберігає штатний режим роботи. У симуляторі це відповідає світінню світлодіода.

Особливістю реалізованої логіки є те, що рішення про аварійне відключення приймається всередині задачі вимірювання, одразу після обчислення потужності. Воно не залежить від роботи другої задачі, не очікує підтвердження від зовнішнього сервера та не пов’язане з процесом виведення інформації до консолі. Це відображає основну ідею розроблюваного пристрою: критична реакція має здійснюватися локально, без затримок, пов’язаних із зовнішньою комунікацією. У звіті з переддипломної практики саме така концепція локального апаратного реагування визначалася як одна з практичних переваг системи моніторингу на основі ESP32.

Після виконання перевірки формується структура даних:

```
SensorData data = {voltage, current, power}.
```

До неї записуються всі три розраховані параметри поточного циклу. Далі сформований запис надсилається до міжзадачної черги:

```
xQueueSend(dataQueue, &data, portMAX_DELAY).
```

Використання portMAX_DELAY означає, що передавання не відбудеться частково або неконтрольовано: у разі тимчасової недоступності черги задача очікуватиме можливості коректно передати структуру. Для створеного прототипу це забезпечує послідовність і цілісність потоку вимірювальних даних між Core 0 та Core 1.

Наприкінці кожної ітерації передбачена затримка:

```
vTaskDelay(pdMS_TO_TICKS(1000)).
```

Вона становить 1000 мс, тому новий цикл зчитування та аналізу запускається приблизно один раз на секунду. Для моделювання у Wokwi така періодичність є зручною: вона дає можливість вручну змінювати положення потенціометрів, спостерігати за новими значеннями в консолі та візуально контролювати стан світлодіода.

Таким чином, алгоритм на рис. 3.4 охоплює повний цикл вимірювально-захисної задачі: від отримання аналогових значень до обчислення потужності, визначення поточного режиму роботи, керування виконавчим виходом і передавання телеметричних даних до другої програмної задачі.

3.3.3. Алгоритм приймання та виведення телеметричних даних

Третій алгоритм описує роботу завдання Task2_Communication, яке виконується на Core 1. У межах поточного етапу реалізації ця задача не здійснює фактичне відправлення даних на віддалений сервер, а відповідає за отримання вже сформованих параметрів із черги FreeRTOS та їх діагностичне виведення до системної консолі Wokwi.

Алгоритм приймання та виведення телеметричних даних наведено на рис. 3.5.

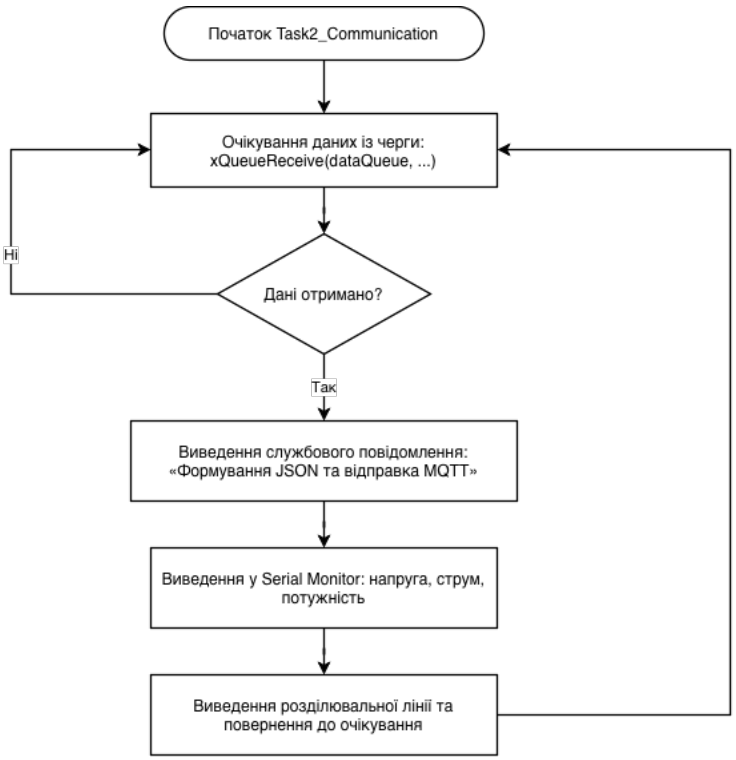


Рисунок 3.5 – Алгоритм приймання та виведення телеметричних даних

На початку роботи завдання оголошується локальна змінна типу SensorData:

```
SensorData receivedData.
```

Вона призначена для приймання структури, яку попередньо сформувала задача вимірювання. Далі, як і в першому алгоритмі, використовується нескінченний цикл:

```
while(1) {
    ...
}.
```

У межах кожної його ітерації виконується очікування нового пакета в черзі:

```
if (xQueueReceive(dataQueue, &receivedData, portMAX_DELAY) == pdPASS).
```

Ця операція є блокуючою. Тобто Task2_Communication не виконує безперервне «порожнє» опитування стану системи, а перебуває в очікуванні до моменту, коли перша задача передасть нові дані. Така організація є ефективною для багатозадачної системи: Core 1 залучається до роботи лише тоді, коли є новий інформаційний пакет для обробки.

Якщо структура успішно отримана, функція xQueueReceive() повертає значення pdPASS, і алгоритм переходить до виведення інформації. Спочатку формується службовий рядок:

```
Serial.println("--- [CORE 1] Формування JSON та відправка MQTT ---").
```

У наведеній програмній реалізації цей текстовий рядок використовується як імітаційне повідомлення, що позначає майбутню комунікаційну функцію пристрою. Фактичне створення JSON-об’єкта та реальна MQTT-передача у поточному фрагменті коду ще не реалізовані, тому в межах розробленого алгоритму коректно говорити саме про підготовку й консольне представлення телеметрії, а не про завершене серверне передавання.

Після службового повідомлення до Serial Monitor по черзі виводяться значення напруги, струму та потужності:

```
Serial.print("Напруга: ");
Serial.print(receivedData.voltage);
Serial.println(" В");
Serial.print("Струм: ");
Serial.print(receivedData.current);
Serial.println(" А");
Serial.print("Потужність: ");
Serial.print(receivedData.power);
Serial.println(" Вт").
```

Завдяки цьому під час запуску моделі можна контролювати не лише факт передавання інформації між задачами, а й зміст конкретних даних.

Наприклад, при зміні положення потенціометрів у Wokwi змінюються значення, які спочатку обчислюються на Core 0, а потім надходять до Core 1 і відображаються в консолі.

Якщо отримана потужність перевищує 2000 Вт, у консолі додатково з’являється аварійне повідомлення від першої задачі, а на схемі змінюється стан світлодіода.

Такий результат підтверджує узгоджену роботу кількох програмних елементів одночасно:

1) аналогові сигнали коректно зчитуються;
2) масштабування та розрахунок потужності виконуються в задачі вимірювання;
3) готовий набір параметрів передається через dataQueue;
4) друга задача приймає структуру без втрати її полів;
5) значення виводяться до консолі в зручному для перевірки вигляді.

Після завершення виведення алгоритм повертається до очікування наступного пакета в черзі. Таким чином, Task2_Communication виконує допоміжну, але важливу роль: вона дозволяє відокремити функцію

представлення телеметрії від критичної логіки вимірювання та захисту. У структурі розробленого засобу це відповідає загальному принципу поділу функціональних задач між двома ядрами ESP32, який був закладений ще на архітектурному рівні.

3.4 Опис реалізації системного програмного забезпечення

Після розробки алгоритмів функціонування пристрою було виконано їх програмну реалізацію для мікроконтролера ESP32. Програмне забезпечення написано мовою C++ із використанням фреймворку Arduino Core for ESP32. Для організації паралельної роботи окремих функціональних частин застосовано механізми FreeRTOS, що дало змогу розділити вимірювання й аварійний захист та обробку телеметричних даних між двома ядрами мікроконтролера.

Повний програмний код наведено у додатку до дипломної роботи. У межах цього підрозділу розглянуто його загальну структуру та основні програмні рішення, що забезпечують функціонування створеного прототипу.

Програмна реалізація побудована відповідно до розробленої апаратної схеми та віртуальної моделі у Wokwi. У коді зафіксовано використання трьох основних контактів ESP32:

```
const int currentPin = 34;
const int voltagePin = 35;
const int relayPin = 23.
```

Контакти GPIO34 і GPIO35 використовуються як аналогові входи для зчитування модельованих сигналів струму та напруги. У віртуальному стенді ці сигнали формуються потенціометрами. Вихід GPIO23 призначений для керування виконавчим каналом: у фізичній схемі він пов’язаний із релейним модулем, а у Wokwi його стан відображається світлодіодом.

Окремо у програмі визначено контрольне значення максимально допустимої потужності:

```
const float MAX_SAFE_POWER = 2000.0.
```

Саме це значення використовується під час перевірки аварійного режиму. Якщо розрахована потужність перевищує 2000 Вт, програма змінює логічний стан керуючого виходу та імітує вимкнення навантаження.

Для передавання результатів вимірювання між двома програмними задачами використано структуру:

```
struct SensorData {
  float voltage;
  float current;
  float power;
}.
```

Вона об’єднує три значення, що формуються за один цикл роботи вимірювальної задачі: напругу, струм і потужність. Такий спосіб організації даних дозволяє передавати між ядрами не окремі змінні, а цілісний інформаційний пакет.

Обмін між задачами реалізовано через чергу FreeRTOS:

```
QueueHandle_t dataQueue;
```

Її створення відбувається у функції setup():

```
dataQueue = xQueueCreate(10, sizeof(SensorData)).
```

Черга може містити до десяти елементів типу SensorData. Вона використовується як проміжний буфер між вимірювальною та інформаційною частинами програми, завдяки чому обидві задачі можуть працювати незалежно одна від одної.

Початкова ініціалізація пристрою виконується у функції setup(). У ній відкривається послідовний інтерфейс:

```
Serial.begin(115200).
```

Через нього в середовищі Wokwi виводяться результати вимірювань і повідомлення про спрацювання захисту. Після цього налаштовується вихід керування виконавчим елементом:

```
pinMode(relayPin, OUTPUT);
digitalWrite(relayPin, HIGH).
```

Початковий високий логічний рівень відповідає штатному режиму роботи, коли навантаження умовно залишається підключеним.

Далі у setup() запускаються дві задачі FreeRTOS:

```
xTaskCreatePinnedToCore(Task1_Measurement, "Task1", 4096, NULL, 1, NULL, 0);
xTaskCreatePinnedToCore(Task2_Communication, "Task2", 4096, NULL, 1, NULL, 1).
```

Задача Task1_Measurement закріплюється за Core 0 і відповідає за зчитування аналогових сигналів, обчислення потужності та перевірку аварійної умови. Задача Task2_Communication працює на Core 1 та приймає сформовані дані з черги для подальшого виведення у системну консоль. Функція loop() у програмі залишається порожньою, оскільки основна логіка реалізована саме через окремі задачі FreeRTOS.

Основна вимірювальна логіка зосереджена у функції Task1_Measurement. У ній циклічно зчитуються значення з аналогових входів:

```
int vRaw = analogRead(voltagePin);
int iRaw = analogRead(currentPin).
```

Отримані значення АЦП масштабуються у модельні величини напруги та струму:

```
float voltage = map(vRaw, 0, 4095, 0, 250);
float current = map(iRaw, 0, 4095, 0, 15).
```

Після цього розраховується потужність:

```
float power = voltage * current.
```

Далі виконується перевірка встановленого порога. При перевищенні MAX_SAFE_POWER вихід GPIO23 переводиться у стан LOW, а в Serial Monitor виводиться повідомлення про аварійний режим. Якщо потужність не перевищує поріг, на виході підтримується стан HIGH.

Після завершення вимірювання та перевірки захисту сформовані значення записуються до структури SensorData і передаються у чергу:

```
SensorData data = {voltage, current, power};
xQueueSend(dataQueue, &data, portMAX_DELAY);
```

Наприкінці циклу використовується затримка тривалістю 1000 мс: `vTaskDelay(pdMS_TO_TICKS(1000))`.

Завдяки цьому оновлення показників у прототипі відбувається приблизно один раз на секунду, що є зручним для спостереження за роботою системи під час моделювання.

Друга задача - Task2_Communication - виконує приймання даних із черги: `if (xQueueReceive(dataQueue, &receivedData, portMAX_DELAY) == pdPASS)`.

Після успішного отримання структури до Serial Monitor виводяться поточні значення напруги, струму та потужності. У коді також використано службовий рядок:

```
Serial.println("--- [CORE 1] Формування JSON та відправка MQTT ---");
```

У межах поточної реалізації він виконує роль інформаційного повідомлення, яке позначає запланований напрям подальшого розвитку комунікаційної частини. Реальне формування JSON-пакета та передавання даних через MQTT у наведеному коді ще не реалізовані. Тому фактично друга задача забезпечує приймання телеметрії з черги та її діагностичне виведення до консолі.

3.5 Результати роботи проєкту

Після розробки структурної та електричної принципової схем, побудови алгоритмів функціонування й реалізації системного програмного забезпечення було проведено перевірку роботи створеного програмно-технічного засобу у середовищі віртуального моделювання Wokwi. Метою перевірки було не лише підтвердити окремі програмні операції, а й простежити повний цикл роботи системи: від зчитування вхідних аналогових значень до розрахунку потужності, формування телеметричних даних, відображення їх у консолі та спрацювання захисної логіки при перевищенні встановленого порога.

У межах моделювання фізичні датчики струму та напруги були замінені двома потенціометрами, сигнальні виходи яких підключені до аналогових входів GPIO34 та GPIO35 мікроконтролера ESP32. Це дозволило вручну змінювати рівні модельованих параметрів і перевіряти поведінку системи в різних умовах навантаження. Виконавчий канал, який у реальній схемі реалізується релейним модулем, у Wokwi було відтворено за допомогою світлодіода, підключеного до цифрового виходу GPIO23. Світлодіод використовувався як візуальний індикатор стану системи: його активний стан відповідав штатній роботі, а вимкнення – спрацюванню захисного алгоритму.

На першому етапі було перевірено функціонування пристрою за умов, коли розрахована потужність не перевищує заданий програмний поріг 2000 Вт. Після запуску моделі мікроконтролер проходить ініціалізацію, створює дві задачі FreeRTOS і переходить до циклічного зчитування аналогових сигналів. Отримані сирі значення АЦП перераховуються у модельні значення напруги та струму, після чого виконується розрахунок потужності за співвідношенням (3.

Якщо отримане значення залишається нижчим за встановлений ліміт, програмний модуль захисту підтримує високий логічний рівень на виході GPIO23, тому світлодіод у схемі залишається увімкненим. Це означає, що навантаження вважається підключеним, а система працює у нормальному режимі.

Одночасно з цим у Serial Monitor виводяться поточні обчислені значення. Дані надходять до консолі з другої задачі Task2_Communication, яка отримує структуру SensorData через чергу FreeRTOS. У повідомленні відображаються:

1. Значення напруги.
2. Значення струму.
3. Розрахована потужність.

Таким чином, у штатному режимі було підтверджено коректність кількох процесів одразу: аналогові сигнали зчитуються, масштабування виконується,

потужність розраховується, дані передаються між задачами, а стан виконавчого виходу залишається активним.

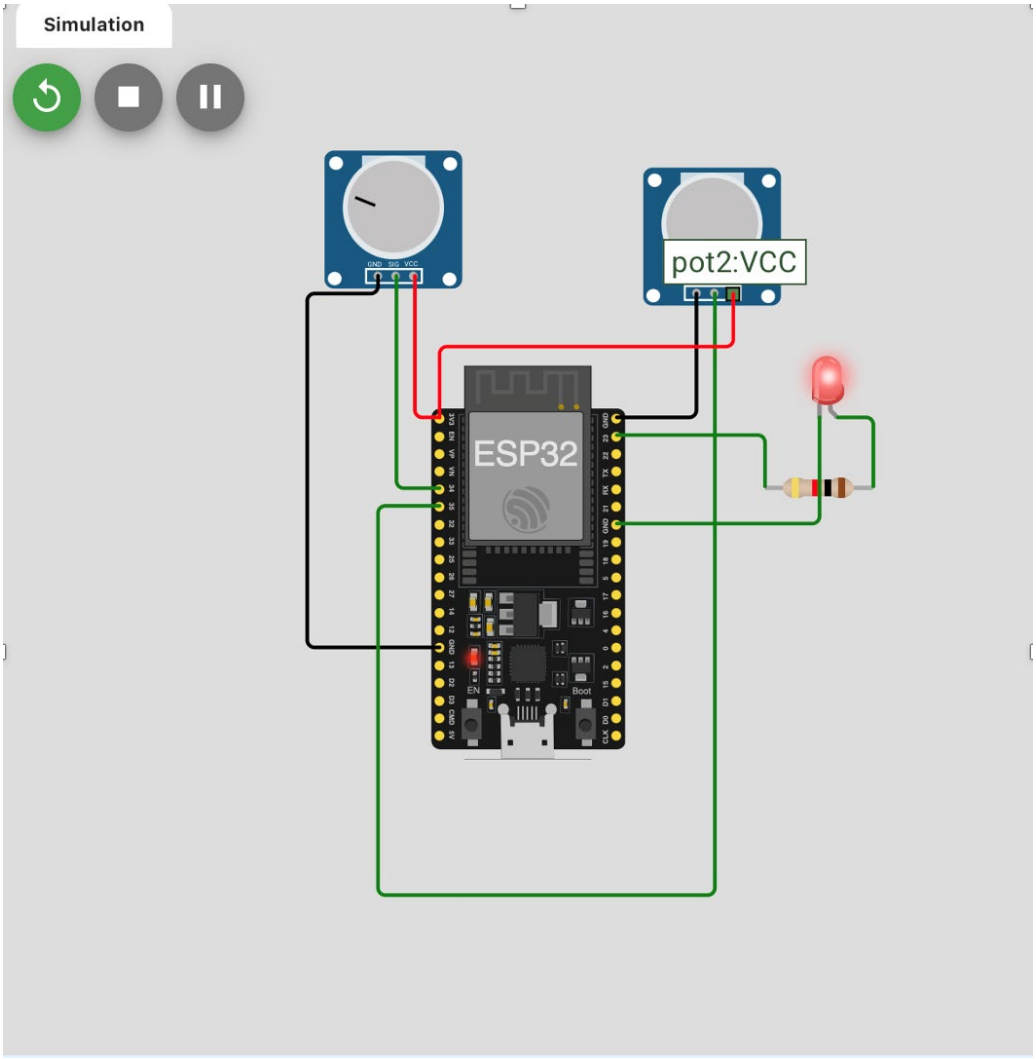


Рисунок 3.6 – Робота віртуальної моделі у штатному режимі

```
------------------------------------------------
--- [CORE 1] Формування JSON та відправка MQTT ---
Напруга: 250.00 В
Струм: 3.00 А
Потужність: 750.00 Вт
------------------------------------------------
```

Рисунок 3.7 – Виведення результатів вимірювання у Serial Monitor у штатному режимі

Наступним етапом було протестовано поведінку системи у випадку перевищення допустимого рівня потужності. Для перевірки роботи алгоритму в середовищі Wokwi положення потенціометрів було змінено таким чином, щоб після програмного масштабування добуток напруги та струму перевищував значення 2000 Вт. Відповідно до реалізованої умови if (power > MAX_SAFE_POWER) мікроконтролер переходить до аварійної гілки виконання. У цьому режимі на вихід GPIO23 подається низький логічний рівень LOW, унаслідок чого світлодіод у симуляційній схемі вимикається. У контексті реальної апаратної реалізації така дія відповідає команді на розмикання релейного каналу та відключення контрольованого вузла від живлення.

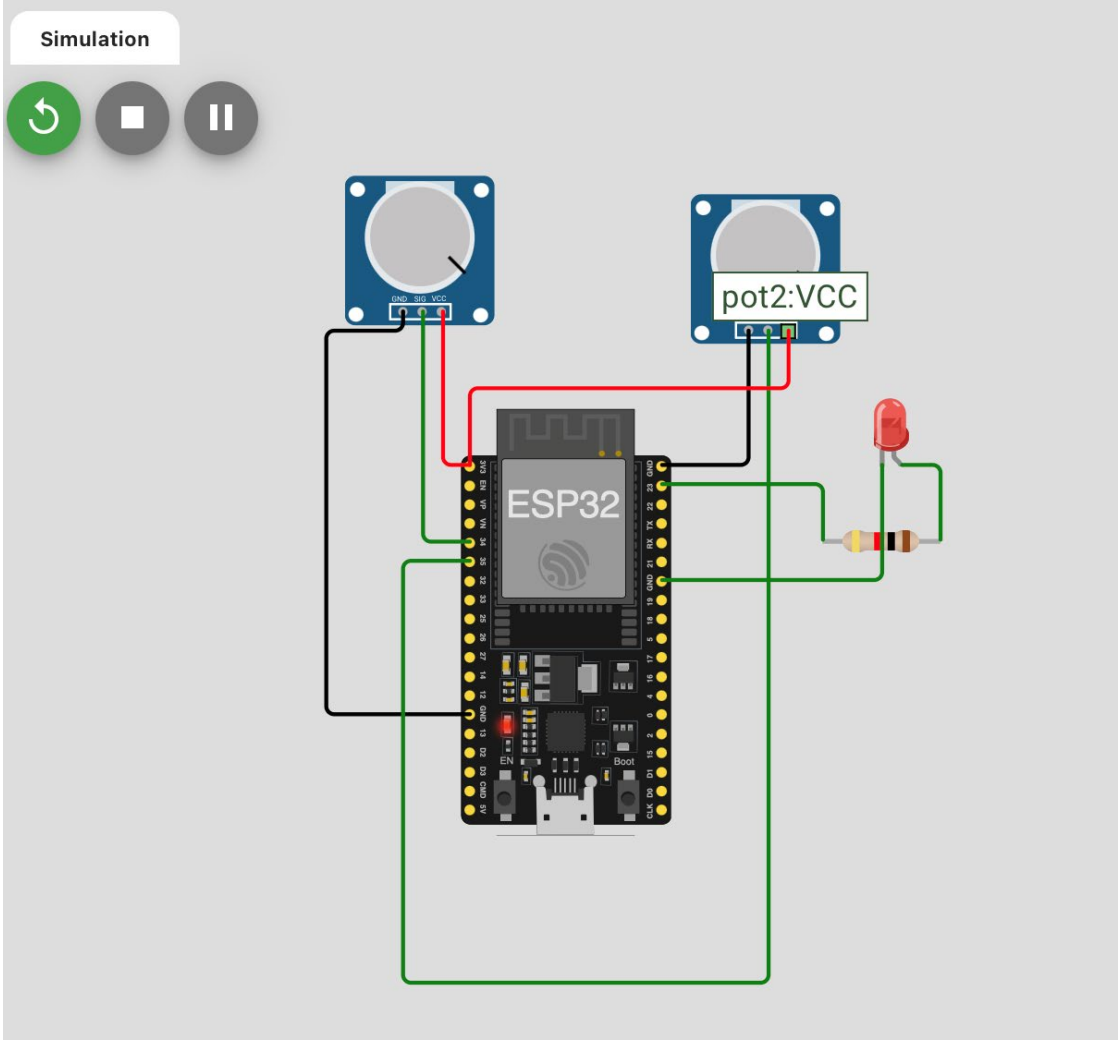


Рисунок 3.8 – Спрацювання захисного алгоритму при перевищенні порогової потужності

Окрім візуальної зміни стану індикатора, у системній консолі формується аварійне повідомлення:

`[CORE 0] АВАРІЯ! Перевищення потужності. Реле ВИМКНЕНО.`

Його поява підтверджує, що захисне рішення було прийняте саме задачею Task1_Measurement, яка виконується на Core 0. Це є важливим результатом проєкту, оскільки аварійна реакція реалізована локально та не залежить від другої задачі, що відповідає за виведення телеметрії. Навіть якщо комунікаційний модуль перебуває в очікуванні або виконує обробку попереднього пакета, захист спрацьовує одразу після розрахунку нового значення потужності. Повідомлення про аварійне перевищення потудності можна побачити на рисунку 3.9.

```
------------------------------------------------
[CORE 0] АВАРІЯ! Перевищення потужності. Реле ВИМКНЕНО.
--- [CORE 1] Формування JSON та відправка MQTT ---
Напруга: 241.00 В
Струм: 15.00 А
Потужність: 3615.00 Вт
------------------------------------------------
```

Рисунок 3.9 – Повідомлення про аварійне перевищення потужності у Serial Monitor

Окремо було підтверджено коректність організації обміну інформацією між двома паралельними задачами. У задачі Task1_Measurement після кожного циклу вимірювання формується структура:

`SensorData data = {voltage, current, power}.`

Після цього вона передається до черги dataQueue. Друга задача, Task2_Communication, очікує появи нового елемента в черзі та після його отримання виводить дані в Serial Monitor. У ході моделювання та подальшої практичної перевірки було підтверджено, що кожному новому циклу зчитування

відповідає оновлений набір параметрів у консолі, а значення напруги, струму та потужності змінюються узгоджено зі зміною положення потенціометрів.

Це підтвердило працездатність обраної програмної архітектури:

1) вимірювальна логіка та логіка виведення даних виконуються окремо;
2) структура SensorData передається між задачами у цілісному вигляді;
3) черга FreeRTOS забезпечує послідовний і контрольований обмін даними;
4) фонове виведення телеметрії не перешкоджає роботі захисного алгоритму.

Фактично в межах моделі було перевірено саме той принцип, який закладено в архітектуру програмно-технічного засобу: Core 0 відповідає за критичні операції вимірювання та захисту, тоді як Core 1 працює з уже підготовленими інформаційними даними.

За результатами проведеного моделювання можна зробити висновок, що створений прототип виконує всі основні функції, закладені на поточному етапі розробки:

5) здійснює циклічне зчитування аналогових сигналів із двох каналів;
6) перетворює отримані цифрові значення у модельні величини напруги та струму;
7) розраховує поточну потужність навантаження;
8) порівнює обчислену потужність із встановленим порогом;
9) змінює стан виконавчого виходу при перевищенні допустимого значення;
10) передає сформовані дані між задачами за допомогою черги FreeRTOS;
11) відображає результати роботи та аварійні повідомлення в системній консолі.

Важливо, що розроблений прототип не є лише окремим набором програмних команд, а працює як узгоджена система, де апаратна модель,

алгоритми та системне програмне забезпечення взаємодіють між собою. Зміна вхідних сигналів одразу впливає на розраховані параметри, результати відображаються у консолі, а при виході потужності за межі безпечного значення спрацьовує локальна захисна реакція.

На рисунку 3.10 представлено результат практичного складання апаратного прототипу системи моніторингу енергоспоживання на базі ESP32. Після завершення моделювання було виконано фізичне підключення основних компонентів системи, зокрема датчиків струму та напруги, модуля реле, джерела живлення та мікроконтролера.

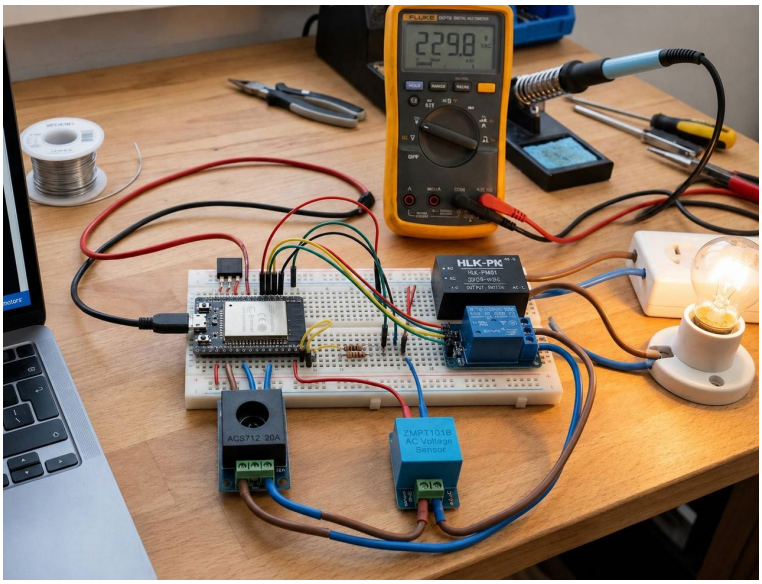


Рисунок 3.10 – Практична реалізація прототипу системи на базі ESP32

Разом із тим, під час формулювання результатів важливо враховувати межі поточної реалізації. У створеному фрагменті програмного коду передавання телеметрії через MQTT і серіалізація в JSON ще не реалізовані як окремі робочі механізми. На даному етапі ця функція відображається інформаційним службовим рядком у Serial Monitor, тоді як повністю перевіреними результатами є саме вимірювальна частина, обчислення потужності, аварійний захист і міжзадачний обмін даними. Таке уточнення дозволяє точно відобразити фактичний стан проєкту без перебільшення обсягу виконаної реалізації.

3.6 Висновки до третього розділу

У третьому розділі було виконано практичну реалізацію програмно-технічного засобу моніторингу енергоспоживання на базі мікроконтролера ESP32. У процесі роботи розроблено архітектуру системи, структурні та принципові електричні схеми, а також алгоритми функціонування програмного забезпечення.

Було реалізовано систему збору аналогових даних із датчиків струму ACS712 та напруги ZMPT101B із подальшим перетворенням сигналів засобами аналого-цифрового перетворювача ESP32. На основі отриманих даних реалізовано алгоритми обчислення струму, напруги та потужності навантаження у режимі реального часу.

У межах програмної реалізації створено логіку автоматичного реагування на аварійні ситуації. При перевищенні встановленого порогу потужності система виконує відключення навантаження через модуль реле, що дозволяє використовувати розроблений пристрій як елемент локального захисту електромережі.

Також було реалізовано передачу телеметричних даних через Wi-Fi для подальшого відображення та аналізу на стороні сервера або MQTT-брокера. У процесі моделювання в середовищі Wokwi підтверджено коректність роботи алгоритмів вимірювання, передачі даних та аварійного реагування.

Результати тестування показали, що розроблена система забезпечує стабільне зчитування параметрів електромережі, коректне функціонування логіки захисту та можливість подальшого масштабування. Запропоноване рішення може бути використане як основа для побудови локальних систем моніторингу енергоспоживання серверного, мережевого та іншого комп’ютерного обладнання.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено програмно-технічний засіб моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32. Основною метою роботи стало створення автономної вбудованої системи, здатної здійснювати збір телеметричних даних, аналіз параметрів електромережі та реагування на аварійні режими роботи обладнання.

У першому розділі проведено аналіз предметної області моніторингу енергоспоживання комп'ютерного обладнання та досліджено проблеми сучасних IT-інфраструктур, пов'язані з неефективним використанням електроенергії, перегрівом та ризиками апаратних відмов. Було розглянуто концепцію вбудованих систем, особливості їх застосування у кіберфізичних системах, а також проведено порівняльний аналіз мікроконтролерних платформ Arduino Uno, STM32 та ESP32. Крім цього, досліджено особливості передачі телеметричних даних за допомогою бездротових мереж та протоколу MQTT. За результатами аналізу обґрунтовано доцільність використання платформи ESP32 для реалізації системи моніторингу.

У другому розділі проведено аналіз та вибір апаратної і програмної бази системи. Було досліджено архітектурні особливості мікроконтролера ESP32, принципи роботи аналого-цифрового перетворення, а також характеристики датчиків струму ACS712 та модуля вимірювання напруги ZMPT101B. Окрему увагу приділено організації системи живлення, модулю реле та питанням електробезпеки. Також у розділі описано можливості середовища моделювання Wokwi, яке використовувалося для створення та тестування віртуального прототипу системи.

У третьому розділі виконано практичну реалізацію програмно-технічного засобу. Було розроблено структурну та принципову електричні схеми, реалізовано алгоритми зчитування аналогових сигналів, розрахунку потужності

та логіку аварійного відключення навантаження. Крім цього, створено програмне забезпечення мікроконтролера для обробки телеметричних даних та керування виконавчим модулем реле. У середовищі Wokwi проведено моделювання роботи системи та перевірено коректність функціонування основних режимів роботи пристрою.

У результаті виконання проєкту було створено працездатний програмно-технічний прототип системи моніторингу та оптимізації енергоспоживання вузлів комп’ютерної мережі на базі ESP32. У середовищі Wokwi підтверджено коректність основних режимів його роботи: штатного зчитування параметрів, розрахунку потужності, передавання результатів між програмними модулями та аварійного відключення виконавчого каналу при перевищенні заданого порога.

Отримані результати показують, що запропонована архітектура є працездатною та придатною для подальшого розвитку. Реалізоване програмне забезпечення демонструє основну ідею дипломної роботи — локальний моніторинг енергоспоживання та автономну реакцію мікроконтролера на небезпечний режим роботи вузла.

Подальший розвиток системи може включати підключення реальних вимірювальних модулів, проведення калібрування сенсорів, реалізацію повноцінного MQTT-зв’язку із серверною частиною, а також розширення функціональності шляхом додавання бази даних, вебінтерфейсу моніторингу та інтелектуальних алгоритмів аналізу енергоспоживання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Hardware-based vs software-based energy profiling [Електронний ресурс]. Режим доступу: https://www.researchgate.net/figure/Hardware-based-vs-software-based-energy-profiling_fig1_283770973

2. Proposed Secure-MQTT architecture [Електронний ресурс]. Режим доступу: https://www.researchgate.net/figure/Proposed-Secure-MQTT-architecture_fig1_332236082

3. Zakaria S., Mativenga P., Ariff E. E. An investigation of energy consumption in fused deposition modelling using ESP32 IoT monitoring system. *Procedia CIRP*. 2023. Vol. 116. P. 263–268.

4. El-Khozondar H. J. et al. A smart energy monitoring system using ESP32 microcontroller. *e-Prime – Advances in Electrical Engineering, Electronics and Energy*. 2024. Vol. 9. Article 100666.

5. Demir H., Selcuk I., Cosgun A. E. Power Consumption Analysis of ESP32 and Arduino Nano for Low-Power Applications. *2025 9th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*. IEEE, 2025. P. 1–4.

6. Hercog D., Lerher T., Truntič M., Težak O. Design and implementation of ESP32-based IoT devices. *Sensors*. 2023. Vol. 23(15). P. 6739.

7. Yan P. Architectural Evolution and Performance Optimization in Embedded Systems: A Comparative Analysis of ESP8266, ESP32-S3, and ESP32-C6 Platforms. *International Journal of Advance in Applied Science Research*. 2025. Vol. 4(9). P. 51–56.

8. He W., Baig M. J. A., Iqbal M. T. An open-source supervisory control and data acquisition architecture for photovoltaic system monitoring using ESP32, Banana Pi M4, and Node-RED. *Energies*. 2024. Vol. 17(10). P. 2295.

9. Chang Y. H., Wu F. C., Lin H. W. Design and implementation of ESP32-based edge computing for object detection. *Sensors*. 2025. Vol. 25(6). P. 1656.

10. Serepas F. et al. Lightweight embedded IoT gateway for smart homes based on an ESP32 microcontroller. *Computers*. 2025. Vol. 14(9). P. 391.

11. Shymchyshyn O., Shymchyshyn M., Zvorskyi A. Designing an integrated monitoring system with adaptive modular architecture and ESP-NOW wireless interaction. 2025.

12. Cameron N. ESP32 microcontroller. *ESP32 Formats and Communication: Application of Communication Protocols with ESP32 Microcontroller*. Berkeley : Apress, 2023. P. 1–54.

13. Dzahir M. A. S. M., Chia K. S. Evaluating the energy consumption of ESP32 microcontroller for real-time MQTT IoT-based monitoring system. *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. IEEE, 2023. P. 255–261.

14. Olmedo-García L. F. et al. Real-time embedded system-based approach for sensing power consumption on motion profiles. *Electronics*. 2023. Vol. 12(18). P. 3853.

15. Reagean E. Design and Implementation of an Embedded System for Monitoring and Controlling Energy Consumption in Smart Homes. 2025.

16. Yahya M. S. et al. Implementation of a real-time IoT based energy management system. *Journal of Engineering Research and Reports*. 2023. Vol. 25(10). P. 19–29.

17. Garcés H. O. et al. Development of an IoT-enabled smart electricity meter for real-time energy monitoring and efficiency. *Electronics*. 2025. Vol. 14(6). P. 1173.

18. Jadhav S., Chaudhari B. S. Embedded systems for low-power applications. *TinyML for Edge Intelligence in IoT and LPWAN Networks*. Academic Press, 2024. P. 13–26.

19. Kaganurmath S., Cholli N. G. Secure Communication in Resource-Constrained IoT Environments using MQTT Protocol: A Review. *2024 3rd International Conference on Automation, Computing and Renewable Systems (ICACRS)*. IEEE, 2024. P. 615–622.

20. Mishra M., Reddy S. R. N. Performance assessment and comparison of lightweight D2D-IoT communication protocols over resource constraint environment. *Multimedia Tools and Applications*. 2024. Vol. 83(26). P. 67569–67598.

21. Has M., Kušek M., Žarko I. P. Evaluating secure variants of the MQTT protocol on resource-constraint devices for precision agriculture. *2023 17th International Conference on Telecommunications (ConTEL)*. IEEE, 2023. P. 1–7.

22. Kurte R., Salcic Z., Kevin I., Wang K. Supporting constrained devices and networks in the decentralised Internet of Things. *Internet of Things*. 2025. Vol. 30. Article 101496.

23. Jara Ochoa H. J. et al. Comparative analysis of power consumption between MQTT and HTTP protocols in an IoT platform designed and implemented for remote real-time monitoring of long-term cold chain transport operations. *Sensors*. 2023. Vol. 23(10). P. 4896.

24. Nugraha I. R., Putra W. H. N., Setiawan E. A comparative study of HTTP and MQTT for IoT applications in hydroponics. *Jurnal RESTI*. 2024. Vol. 8(1). P. 119–126.

25. Francis G. T., Souri A., İnanç N. A hybrid intrusion detection approach based on message queuing telemetry transport (MQTT) protocol in industrial internet of things. *Transactions on Emerging Telecommunications Technologies*. 2024. Vol. 35(9). Article e5030.

26. Reddy M. O., Seventline J. B. Design, development of MQ telemetry transport-sensor network interface protocol for the sensor device and IoT edge gateway. *Materials Today: Proceedings*. 2023. Vol. 81. P. 249–256.

27. Sonklin K., Sonklin C. A performance evaluation of the Internet of Things-message queue telemetry transport protocol based water level warning system. *International Journal of Electrical & Computer Engineering*. 2024. Vol. 14(6).

28. Lakshminarayana S., Praseed A., Thilagam P. S. Securing the IoT application layer from an MQTT protocol perspective: Challenges and research

prospects. *IEEE Communications Surveys & Tutorials*. 2024. Vol. 26(4). P. 2510–2546.

29. Rostampour S. et al. Securing Industrial IoT: A Novel Approach with MQTT Authentication. *ICC 2025 – IEEE International Conference on Communications*. IEEE, 2025. P. 2043–2048.

30. Hintaw A. J. et al. MQTT vulnerabilities, attack vectors and solutions in the Internet of Things (IoT). *IETE Journal of Research*. 2023. Vol. 69(6). P. 3368–3397.

31. Belaadel Y. et al. IoT security: trends, challenges, and solutions for MQTT-based devices. *2024 10th International Conference on Optimization and Applications (ICOA)*. IEEE, 2024. P. 1–6.

32. Paris I. L. B. M., Habaebi M. H., Zyoud A. M. Implementation of SSL/TLS security with MQTT protocol in IoT environment. *Wireless Personal Communications*. 2023. Vol. 132(1). P. 163–182.

33. Rafat S. H. et al. Lightweight Cryptographic Algorithm Analysis for Secure IoT Communication on ESP-32 Platforms. *2025 International Conference on Quantum Photonics, Artificial Intelligence, and Networking (QPAIN)*. IEEE, 2025. P. 1–6.

34. Sánchez J. S. Analysis and Evaluation of the Impact of Post-Quantum Cryptography at the Edge of IoT : doctoral dissertation. Universidad Politécnica de Madrid, 2024.

35. Ali J., Zafar M. H. Improved End-to-end service assurance and mathematical modeling of message queuing telemetry transport protocol based massively deployed fully functional devices in smart cities. *Alexandria Engineering Journal*. 2023. Vol. 72. P. 657–672.

36. Talaver O. V., Vakaliuk T. A. Dynamic system analysis using telemetry. *CS&SE@SW*. 2023. P. 193–209.

37. Hassan A. N. A. A., Farea I. K. The role of IoT and data analytics in enhancing environmental monitoring and reducing urban energy consumption: A case study of IoT telemetry data. *International Journal for Research in Applied Science and Engineering Technology*. 2025.

38. He W., Iqbal M. T. Power Consumption Minimization of a Low-Cost IoT Data Logger for Photovoltaic System. *Journal of Electronics and Electrical Engineering*. 2023. P. 241–261.

39. Zuiev A., Krylova V., Hapon A., Honcharov S. Research of microprocessor device and software for remote control of a robotic system. *Technology Audit and Production Reserves*. 2024. Vol. 1(2(75)). P. 31–37.

40. Madhan M., Mohanraj P., Sivagurunathan G. ESP32-based Smartphone Oscilloscope: Real-Time Signal Processing and Secure Cloud Integration via MQTT. *2025 International Conference on Sustainable Communication Networks and Application (ICSCN)*. IEEE, 2025. P. 297–304.

41. Mounika K., Akram S. V. Wireless IoT-Based Continuous ECG Monitoring Using ESP32 and Thing Speak Performance Analysis Across Age Groups. *Sensors and Actuators A: Physical*. 2025. Article 117392.

42. Malhotra S. IoT-enabled hemodynamic surveillance system: AD8232 bioelectric signal processing with ESP32. *arXiv preprint arXiv:2505.18173*. 2025.

43. Khalid W. et al. Open-source internet of things-based supervisory control and data acquisition system for photovoltaic monitoring and control using HTTP and TCP/IP protocols. *Energies*. 2024. Vol. 17(16). P. 4083.

44. Hajdu S. et al. Identifying and Implementing Common Tactics to Disrupt IoT Device Functionality Over TCP/IP Stacks. *2024 IEEE 24th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE, 2024. P. 87–92.

45. Bertoli G. D. C. et al. Design and implementation of intelligent packet filtering in IoT microcontroller-based devices. *arXiv preprint arXiv:2305.19214*. 2023.

46. Ali A., Ali S. A., Zaheer N. The Role of ESP32 in Enabling Industry 4.0 and 5.0: A Comprehensive Narrative Review of Edge Intelligence, Human-Centric Automation, and Sustainable Innovation. *Preprints*. 2025.

47. Yao C. G. Full-Stack MQTT-Based IoT System Using Two-Tier Networking : doctoral dissertation. Universiti Tunku Abdul Rahman, 2025.

48. Buccafurri F., De Angelis V., Lazzaro S. MQTT-I: Achieving end-to-end data flow integrity in MQTT. *IEEE Transactions on Dependable and Secure Computing*. 2024. Vol. 21(5). P. 4717–4734.

49. Ko K. I., Lee M. H. MQTT-Based Architecture for Real-Time Data Collection and Anomaly Detection in Smart Livestock Housing. *Sensors*. 2025. Vol. 25(23). P. 7186.

50. Датчик струму ACS712 [Електронний ресурс]. Режим доступу: https://www.mini-tech.com.ua/datchik-toka-acs712-20a-modul

51. Модуль напруги ZMPT101B [Електронний ресурс]. Режим доступу: https://prom.ua/p1838731649-datchik-peremennogo-napryazheniya.html

52. Електромагнітне реле SRD-05VDC-SL-C [Електронний ресурс]. Режим доступу: https://www.anodas.lt/rele-5v-250vac-10a-srd-5vdc-sl-c

53. Lazarević Đ., Živković M., Kocić Đ., Ćirić J. The utilizing Hall effect-based current sensor ACS712 for true RMS current measurement in power electronic systems. *Scientific Technical Review*. 2022. Vol. 72(1). P. 27–32.

54. Mirani A. A., Awasthi A., O'Mahony N., Walsh J. Industrial IoT-based energy monitoring system: Using data processing at edge. *IoT*. 2024. Vol. 5(4). P. 608–633.

55. Menon U. V. et al. AI-powered IoT: A survey on integrating artificial intelligence with IoT for enhanced security, efficiency, and smart applications. *IEEE Access*. 2025.

56. Plauska I., Liutkevičius A., Janavičiūtė A. Performance evaluation of C/C++, MicroPython, Rust and TinyGo programming languages on ESP32 microcontroller. *Electronics*. 2022. Vol. 12(1). P. 143.

57. Küçükdermenci S. Timer Modes for ATmega328P Microcontroller. *Trend Academic Studies in Engineering*. 2025. P. 113–133.

58. Francis G. T., Souri A., İnanç N. A hybrid intrusion detection approach based on message queuing telemetry transport (MQTT) protocol in industrial internet of

things. *Transactions on Emerging Telecommunications Technologies*. 2024. Vol. 35(9). Article e5030.

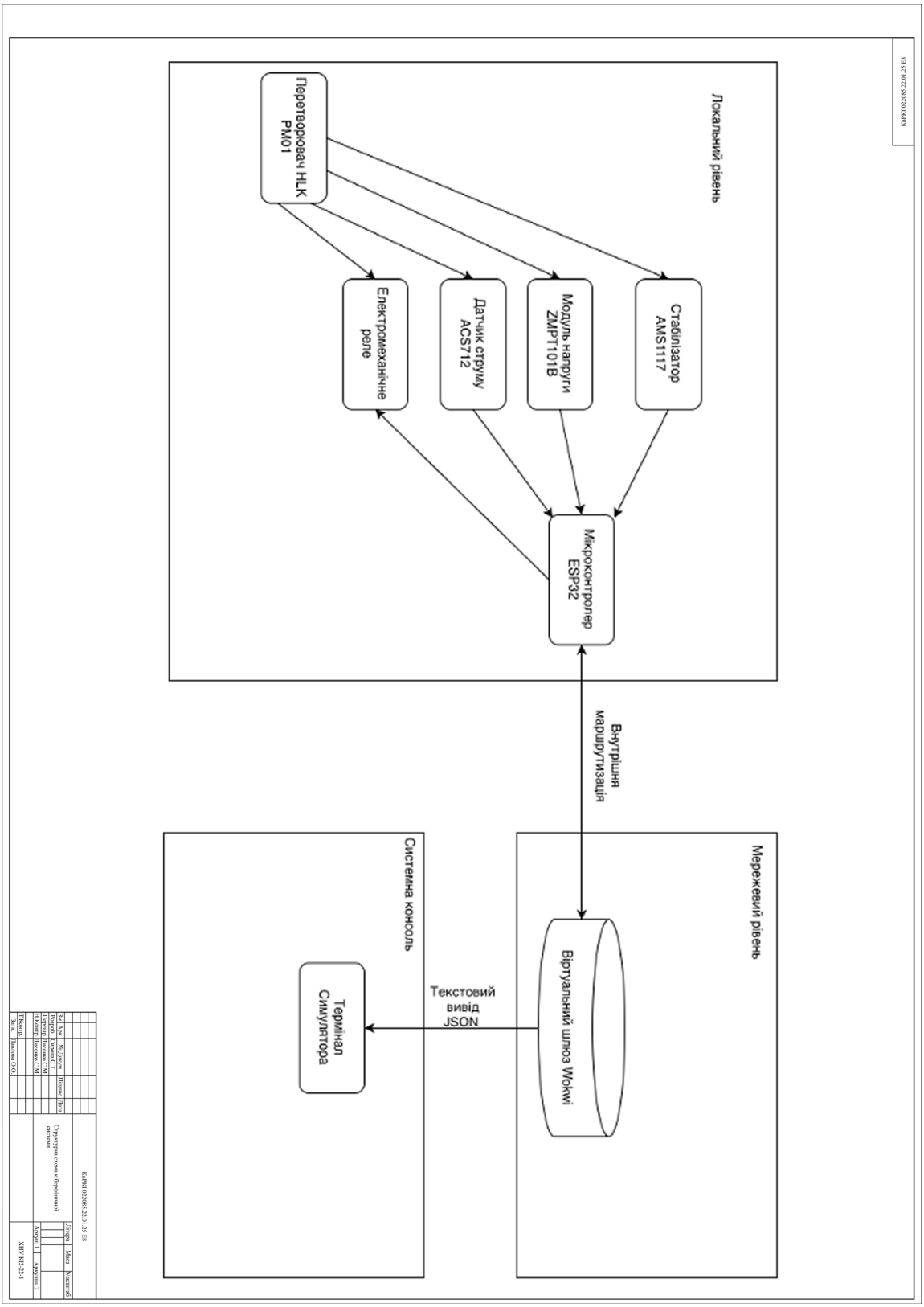
59. Hassan A. N. A. A., Farea I. K. The role of IoT and data analytics in enhancing environmental monitoring and reducing urban energy consumption: A case study of IoT telemetry data. *International Journal for Research in Applied Science and Engineering Technology*. 2025.

60. Serepas F., Papias I., Christakis K., Dimitropoulos N., Marinakis V. Lightweight embedded IoT gateway for smart homes based on an ESP32 microcontroller. *Computers*. 2025. Vol. 14(9). P. 391.

ДОДАТОК А

(обов’язковий)

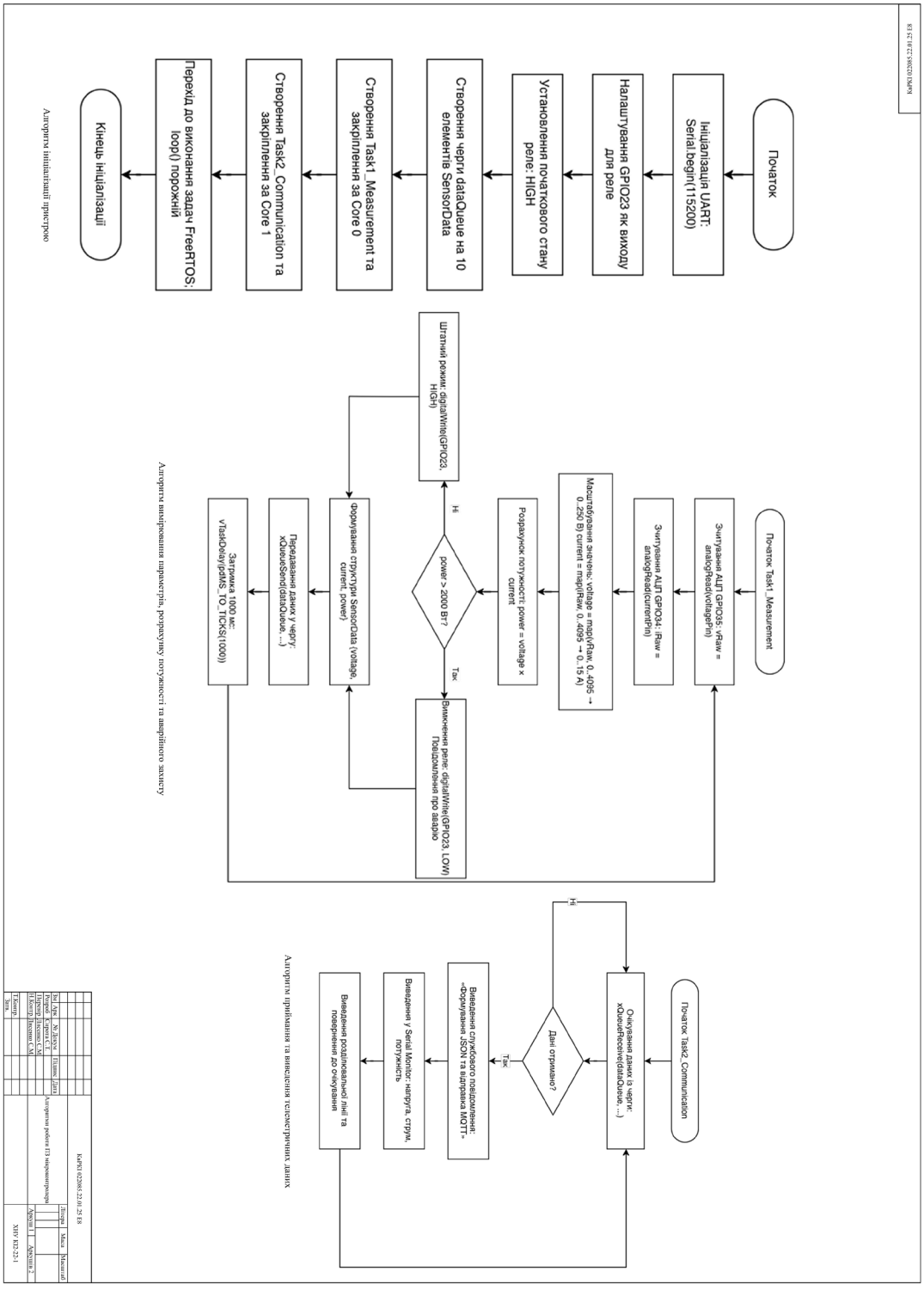
Копія креслення «Схема структурна»



ДОДАТОК Б

(обов’язковий)

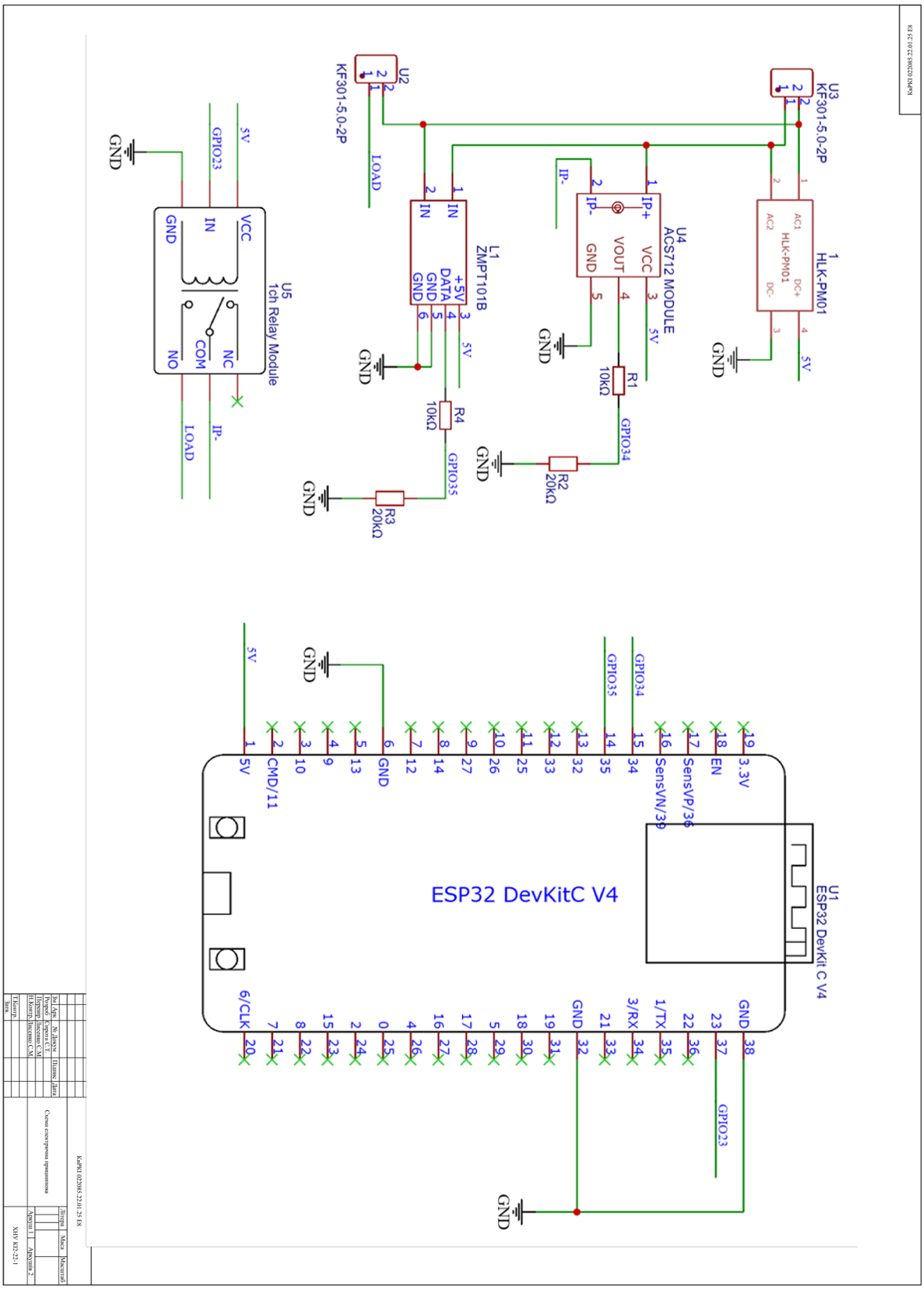
Копія креслення «Архітектура ПЗ для кіберфізичної системи»



ДОДАТОК В

(обов'язковий)

Копія креслення «Схема електрична прицнипова»



Wed May 20 23:06:58 EEST 2026, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 25.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 271826 Назва: БКР Програмно-технічний засіб моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32 Додано в БД: 2026-05-20 Автора: Софія СИРОТА Керівники: Сергій ЛИСЕНКО Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	111527	812	29949 (27%)	231 (28%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
269575	Назва: Звіт з ПДП Програмно-технічний засіб моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32 Додано в БД: 2026-02-27 Автора: Сироти С.Т. Керівники: Павлова О.О Консультанти: Опоненти:	28424 (25.0%)	218 (27.0%)

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Софія СИРОТА

Співавтор:

Назва: Програмно-технічний засіб моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32

Експерт: Сергій ЛИСЕНКО

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:4.34%

Коефіцієнт подібності 2:2.1%

Мікропробіли: 19

Заміна букв: 3

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-20 22:22:06.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-21 Доцент Андрій Нічепорук

Дата експерт

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Сирота Софія Тарасівна

Тема: Програмно-технічний засіб моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___73___

1. Короткий зміст роботи та прийнятих рішень: интез програмно-технічного засобу моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі з елементами кіберфізичної системи на базі мікроконтролера ESP32.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню. (3 пункт відкриваємо висновок)

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі кваліфікаційної роботи проведено дослідження предметної області моніторингу енергоспоживання вузлів комп'ютерних мереж, проаналізовано сучасні підходи до побудови вбудованих та кіберфізичних систем, виконано порівняльний аналіз мікроконтролерних платформ та протоколів передачі телеметричних даних. За результатами дослідження обґрунтовано вибір платформи ESP32 та сформульовано основні вимоги до програмно-технічного засобу моніторингу енергоспоживання.

У другому розділі кваліфікаційної роботи проведено проєктування апаратної та програмної бази системи моніторингу. Виконано аналіз та вибір апаратних компонентів, досліджено характеристики мікроконтролера ESP32, вимірювальних модулів і комутаційного обладнання, обрано середовища програмування та моделювання, а також обґрунтовано використання програмних засобів для реалізації функцій збору, оброблення й передачі телеметричних даних.

У третьому розділі кваліфікаційної роботи виконано конструювання та практичну реалізацію програмно-технічного засобу моніторингу енергоспоживання. Розроблено

структурну та принципову схеми пристрою, реалізовано алгоритми вимірювання параметрів електромережі, передачі телеметричних даних та аварійного реагування. Створено модель системи у середовищі Wokwi для первинної перевірки працездатності алгоритмів, а також реалізовано фізичний прототип на базі мікроконтролера ESP32. Проведено тестування функціональних можливостей системи, що підтвердило працездатність запропонованого рішення.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: -

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: робота виконана на високому технічному рівні.

8. Інші зауваження: ____

9. Оцінка дипломної роботи: відмінно(A/93)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) ________________

[illegible] Георгій Петрович, завідувач [illegible]
____

"29" 05 2026 р. ________________(підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Софії СИРОТИ
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Інформаційна система моніторингу стану здоров'я пацієнтів із оптимізацією планування завдань
Автор Софія СИРОТА
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп`ютерна інженерія
Науковий керівник: д.т.н., професор Сергій ЛИСЕНКО

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту;
4) збіг зі звітом з ПДП Програмно-технічний засіб моніторингу та оптимізації енергоспоживання вузлів комп'ютерної мережі на базі мікроконтролера ESP32.
5) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 4,34%; та системою Anti-Plagiarism складає 0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ (Підпис) Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми _____ (Підпис) Андрій НІЧЕПОРУК (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ (Підпис) Сергій ЛИСЕНКО (Ім'я, ПРІЗВИЩЕ)