Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра обчислювальної техніки
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«УНІВЕРСАЛЬНИЙ МІКРОПРОЦЕСОРНИЙ МОДУЛЬ ДЛЯ IoT ЗАСТОСУВАНЬ»

Виконав: студент 2-го курсу, групи 1КІ-24м
спеціальності 123 «Комп'ютерна інженерія »
(шифр і назва напряму підготовки, спеціальності)

Ткаченко Р.В.
(прізвище та ініціали)

Керівник: к. т. н., доц. каф. ОТ

Тарновський М.Г.
(прізвище та ініціали)
«12» 12 2025 р.

Опонент: к. т. н., доц. каф. ПЗ

Черноволик Г. О.
(прізвище та ініціали)
«12» 12 2025 р.

Допущено до захисту
Завідувач кафедри ОТ

д.т.н., проф. Азаров О. Д.

«17» 12 2025 р.

Вінниця ВНТУ- 2025 рік

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Рівень вищої освіти ІІ-й (магістерський)

Галузь знань – 12 «Інформаційні технології»

Спеціальність – 123 « Комп'ютерна інженерія »

Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ, д.т.н., проф.

_____ Азаров О. Д.

" 18 " вересня 2025 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

студенту Ткаченку Роману Віталійовичу

1 Тема роботи: «Універсальний мікропроцесорний модуль для IoT застосувань», керівник роботи к.т.н., доц. каф. ОТ Тарновський М.Г, затверджено наказом ВНТУ №313 від 24.09.2025 р.

2 Термін подання студентом роботи 17.12.2025 р

3 Вихідні дані до роботи: призначення — апаратно-програмна платформа для розробки пристроїв IoT; інтерфейс підключення до мережі Інтернет — WiFi; кількість підтримуваних сенсорів — не менше 3; кількість каналів управління — не менше 2; живлення — від зовнішнього джерела постійної напруги та від акумулятора.

4 Зміст пояснювальної записки: вступ, аналіз предметної області, аналіз теоретичних основи архітектур та протоколів IoT-систем, розробка апаратно-програмних засобів модуля, мікропроцесорного модуля, економічна частина.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): структурна схема універсального мікропроцесорного модуля, функціональна схема універсального мікропроцесорного модуля, блок-схема

програмного забезпечення універсального мікропроцесорного модуля, блок-схема алгоритму реєстрації універсального мікропроцесорного модуля у застосунку.

6 Консультанти розділів роботи представлені у таблиці 1

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–4	Тарновський М.Г., к. т. н., доц. каф. ОТ	25.09.25	15.12.25
5	Ратушняк О.Г., к.т.н., доц. каф. ЕПВМ		
Нормоконтроль	Швець С.І., асист. каф. ОТ		

7. Дата видачі завдання 25.09.2025 р

8. Календарний план виконання МКР наведений в таблиці 2

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Примітка
1	Постановка задачі роботи	08.09.2025	виконано
2	Обгрунтування актуальності теми. Аналіз технологічних рішень	20.09.2025	виконано
3	Аналіз теоретичні основи архітектур та протоколів іот-систем	27.09.2025	виконано
4	Розробка апаратно-програмних засобів модуля	30.09.2025	виконано
5	Розробка рекомендацій з експлуатацію та налаштування	15.10.2025	виконано
6	Оцінка комерційного потенціалу розробки	17.10.2025	виконано
7	Оформлення пояснювальної записки та ілюстративного матеріалу	04.11.2025	виконано
8	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	10.11.2025	виконано

Студент _____ Ткаченко Р.В.
(підпис)

Керівник роботи _____ Тарновський М.Г.
(підпис)

АНОТАЦІЯ

УДК 004.9

Ткаченко Р.В. Універсальний мікропроцесорний модуль для IoT застосувань. Магістерська кваліфікаційна робота зі спеціальності 123 – Комп’ютерна Інженерія,. Вінниця: ВНТУ, 2025. 105 с.

На укр. мові. Бібліогр.: назв 40; рис.: 31; табл.12.

У магістерській кваліфікаційній роботі розглянуто питання розробки універсального мікропроцесорного модуля для застосувань Інтернету речей. У роботі проведено аналіз сучасних архітектур IoT-систем, протоколів зв’язку та засобів забезпечення безпеки, досліджено особливості реалізації багатозадачних енергоефективних вбудованих систем. Розроблено апаратну частину модуля на базі платформи ESP32-S3 з інтегрованими бездротовими інтерфейсами Wi-Fi та Bluetooth LE, підсистемою збору даних на основі сенсорів BME280 та BH1750, модулем реального часу DS3231 та вузлом керування зовнішніми виконавчими елементами. Систему живлення спроєктовано з використанням літій-полімерного акумулятора, зарядного контролера та стабілізатора 3.3 В, що забезпечує автономну роботу модуля. Програмне забезпечення реалізовано у середовищі FreeRTOS із підтримкою багатопотокової обробки, збору та передавання даних, а також захищеного обміну з хмарною інфраструктурою через MQTT-протокол.

Ключові слова: Інтернет речей, мікропроцесорний модуль, ESP32-S3, FreeRTOS, MQTT, Wi-Fi, BLE.

ABSTRACT

Ткаченко R.V. Universal Microprocessor Module for IoT Applications. Master's Qualification Thesis in Specialty 123 – Computer Engineering. Vinnytsia: VNTU, 2025. 105 p.

In the Ukrainian language. Bibliogr. : 40 titles; 31 figures; 12 tables.

The master's qualification thesis addresses the issues of developing a universal microprocessor-based module for Internet of Things applications. The work includes an analysis of modern IoT system architectures, communication protocols, and security mechanisms, as well as a study of the principles of implementing multitasking, energy-efficient embedded systems. The hardware of the module was developed on the basis of the ESP32-S3 platform with integrated wireless interfaces Wi-Fi and Bluetooth LE, a data acquisition subsystem based on the BME280 and BH1750 sensors, a DS3231 real-time clock module, and a control unit for external actuators. The power supply system was designed using a lithium-polymer battery, a charging controller, and a 3.3 V voltage regulator, ensuring autonomous operation of the module. The software was implemented in the FreeRTOS environment with support for multithreaded processing, data acquisition and transmission, as well as secure communication with cloud infrastructure via the MQTT protocol.

Keywords: Internet of Things, microprocessor module, ESP32-S3, FreeRTOS, MQTT, Wi-Fi, BLE.

ЗМІСТ

ВСТУП

Сучасний етап розвитку суспільства характеризується стрімкою цифровізацією та глибокою інтеграцією технологій у всі сфери життєдіяльності людини. Інформаційні та телекомунікаційні рішення перетворилися на базову інфраструктуру суспільства, забезпечуючи автоматизацію, аналітику та обмін даними на глобальному рівні. Вагому роль у цьому процесі відіграє Інтернет речей (Internet of Things, IoT) — технологічна концепція, що об’єднує пристрої, сенсори та системи управління в єдине інформаційне середовище. Це надає звичайним пристроям нові можливості, збагачуючи наше повсякденне життя більш широкою функціональністю та новим рівнем автоматизації.

Актуальність дослідження зумовлена експоненціальним зростанням парку IoT-пристроїв та виникненням нагальної потреби у створенні спеціалізованих мікропроцесорних засобів, які б поєднували високу продуктивність, широку функціональність, енергоефективність, компактність та інтегровані засоби комунікації. Зростаюча популярність засобів Інтернету речей створює попит на універсальні багатофункціональні апаратно-програмні платформи, використання яких дозволяє спростити розробку та впровадження кінцевого продукту IoT.

Метою роботи є розширення функціональних можливостей мікропроцесорного модуля для IoT застосувань за рахунок апаратно-програмної інтеграції у ньому засобів для вимірювання температури, вологості, освітленості та тиску.

Для досягнення поставленої мети у роботі вирішуються такі **задачі**:

— аналіз сучасних технологій в області Інтернету речей;

— аналіз архітектур мікропроцесорних систем та протоколів обміну даними;

— розробка апаратної частини універсального модуля для IoT застосувань;

— розробка програмних засобів універсального модуля для IoT застосувань;

— оцінка економічної ефективності впровадження запропонованого технічного рішення.

Об'єктом дослідження є інформаційні процеси, що пов'язані з функціонуванням пристроїв Інтернету речей.

Предметом дослідження є апаратні та програмні засоби Інтернету речей.

Методи дослідження базуються на методах порівняльного аналізу та схемотехнічного проектування.

Новизна роботи полягає в тому, що набула подальшого розвитку концепція універсальної апаратно-програмної платформи, в якій за рахунок апаратно-програмної інтеграції додаткових електронних компонентів забезпечена можливість її використання для вирішення різних задач в IoT застосуваннях без додаткового доопрацювання.

Практичне значення полягає у можливості безпосереднього використання розробленого апаратно-програмного комплексу як готового рішення для створення IoT-пристроїв різного призначення, що дозволяє скоротити часові та фінансові витрати на етапі проектування.

Апробація результатів магістерської кваліфікаційної роботи здійснена в доповіді на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)» [1]:

Ткаченко Р.В., Тарновський М.Г. Універсальний мікропроцесорний засіб для IoT застосувань Конференція ВНТУ: Молодь в науці: дослідження, проблеми, перспективи (МН-2026). Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26630

1 АНАЛІЗ ТЕХНОЛОГІЧНИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Сучасний стан, архітектурні моделі та проблеми Інтернету речей

Сучасний етап розвитку інформаційних технологій характеризується фундаментальним зсувом від простої автоматизації до створення глобальних, інтегрованих кіберфізичних систем. Центральною концепцією цього переходу є Інтернет речей (Internet of Things, IoT). Це парадигма обчислювальної мережі, що складається з унікально ідентифікованих фізичних об’єктів («речей»), які оснащені вбудованими сенсорами, мікропроцесорами, виконавчими механізмами та засобами комунікації. Ці пристрої здатні збирати дані з навколишнього середовища, взаємодіяти один з одним (M2M) та обмінюватися даними із зовнішніми системами через стандартизовані протоколи зв'язку, часто без безпосереднього втручання людини [2, 3].

Історично, термін IoT був запропонований Кевіном Ештоном у 1999 році, перш за все, для опису використання технології RFID в оптимізації логістичних ланцюгів. Однак сьогодні ця ідея трансформувалася в глобальну екосистему, що охоплює всі сфери життя, від побутових приладів до критичної промислової інфраструктури. За прогнозами аналітичних агенцій, кількість підключених IoT-пристроїв перевищить 29 мільярдів одиниць до 2030 року, генеруючи при цьому зетабайти даних [2].

Розвиток IoT відбувається за двома основними векторами: промисловий IoT (IoT), що фокусується на автоматизації виробництва (Industry 4.0), предиктивному обслуговуванні обладнання, логістиці та агросекторі, та споживчий IoT, що охоплює "розумні" будинки, носиму електроніку та системи персонального моніторингу здоров'я. Обидва напрямки висувають жорсткі, хоч і дещо різні, вимоги до апаратних вузлів.

Ключова відмінність сучасної концепції IoT від раніших систем M2M полягає у зміщенні акценту з простої телеметрії на інтелектуальний аналіз зібраних даних та прийняття автономних рішень. Для обробки цих величезних

масивів даних (Big Data) склалися дві основні архітектурні моделі: централізована (хмарна) та децентралізована (туманна/гранична). Візуалізація хмарної моделі, з периферійними вузлами, наведена на рисунку 1.1.

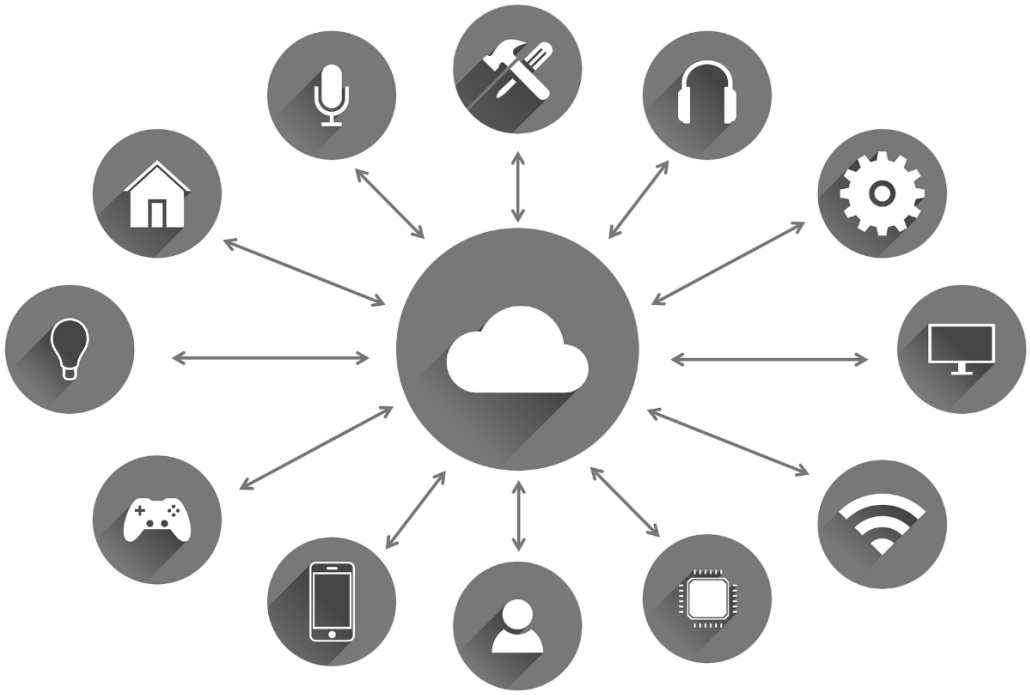


Рисунок 1.1 — Концептуальна схема об'єднання різнорідних пристроїв у єдину хмарну екосистему Інтернету речей

Централізована модель хмарних обчислень (Cloud Computing) на початковому етапі розвитку IoT передбачала виконання більшості обчислень на стороні хмари, тоді як кінцеві пристрої здійснювали лише базові функції збору даних. Їхнє завдання — зібрати «сирі» дані та відправити їх через Інтернет на потужні віддалені сервери (в "хмару") [4]. Хмарні платформи (такі як AWS IoT, Google Cloud Platform, Microsoft Azure) надають практично необмежені, еластичні ресурси для довгострокового зберігання, складної аналітики, тренування моделей машинного навчання та глобальної візуалізації даних. Незважаючи на потужність, цей підхід має фундаментальні недоліки для багатьох застосунків. По-перше, це висока мережева затримка (latency), оскільки дані мають подолати шлях до сервера і назад, що є неприйнятним для

систем, що вимагають реакції в реальному часі (наприклад, промислова автоматика або медичні прилади). По-друге, це високі вимоги до пропускної здатності каналу та значні витрати трафіку, оскільки всі дані, навіть несуттєві, передаються на сервер.

Децентралізована модель туманних (Fog) та граничних (Edge) обчислень виникла у відповідь на обмеження хмарної архітектури та передбачає перенесення обчислювальних процесів ближче до джерел даних [5]. Граничні обчислення (Edge Computing) передбачають, що аналіз, фільтрація та прийняття рішень виконуються безпосередньо на самому кінцевому пристрої (наприклад, на мікроконтролері ESP32). Туманні обчислення (Fog Computing) є проміжним шаром, де обробку виконує локальний шлюз (наприклад, роутер або спеціалізований міні-сервер), який збирає дані з групи Edge-пристроїв. Такий підхід дозволяє забезпечити миттєву реакцію системи на події (наприклад, модуль сам вирішує увімкнути MOSFET-ключ при падінні освітленості, не чекаючи команди з хмари), значно знизити навантаження на мережу (у хмару відправляються лише агреговані звіти або тривожні сповіщення) та забезпечити базове функціонування навіть за повного зникнення інтернет-з'єднання [5]. Саме ця модель є найбільш перспективною для розробки універсальних та надійних IoT-модулів.

Незважаючи на стрімкий розвиток, залишаються дві фундаментальні проблеми, що стримують масове та безпечне впровадження IoT-рішень проблема сумісності (Interoperability) та проблема безпеки.

Ринок IoT є вкрай фрагментованим. Існує величезна кількість конкуруючих стандартів та протоколів зв'язку (Wi-Fi, Bluetooth LE, Zigbee, Z-Wave, LoRaWAN), які часто є несумісними між собою. Це створює гетерогенне середовище ("зоопарк протоколів"), в якому пристрої від різних виробників не можуть взаємодіяти один з одним без складних програмних або апаратних шлюзів [3, 6]. Це змушує розробників універсальних модулів або обирати один стандарт, обмежуючи ринок, або проектувати складні гібридні пристрої, здатні працювати в кількох мережах одночасно. Ключові компоненти та виклики, що

характеризують сучасний етап розвитку Інтернету речей, наведено на рисунку 1.2.

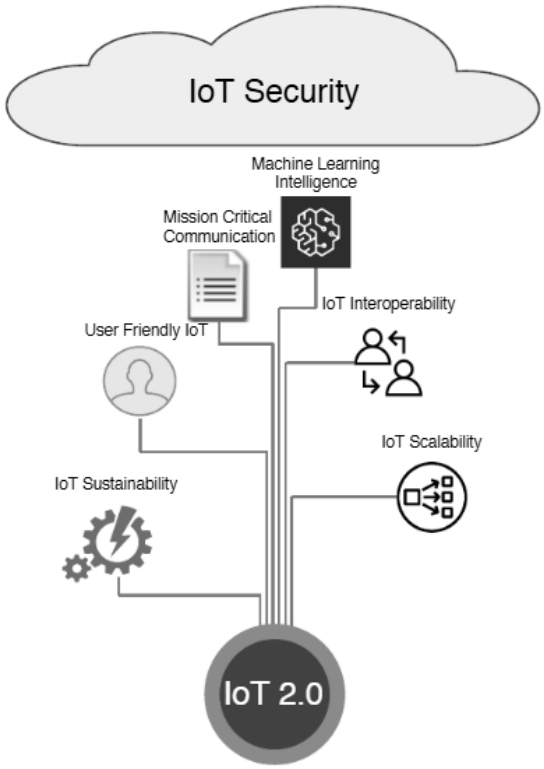


Рисунок 1.2 — Ключові вимоги та компоненти IoT: безпека, сумісність та інтелект [7].

Безпека є, можливо, найголовнішим викликом. Кожен підключений IoT-пристрій є потенційною точкою вразливості в мережі [7].

Проблеми безпеки можна розділити на кілька рівнів:

— конфіденційність — передавання даних у відкритому вигляді (наприклад, із використанням протоколів HTTP або MQTT без шифрування) створює можливість їх перехоплення зловмисниками шляхом пасивного прослуховування мережевого трафіку;

— цілісність — за відсутності механізмів захисту зловмисник може не лише здійснювати несанкціоноване зчитування даних, але й модифікувати їх у процесі передавання, формуючи та надсилаючи на сервер підроблені показники;

— автентифікація — серверна сторона повинна мати засоби перевірки автентичності клієнтського пристрою з метою запобігання ін’єкції даних від неавторизованих або клонованих вузлів, тоді як клієнт повинен автентифікувати сервер для уникнення підключення до зловмисної інфраструктури в межах атаки типу «людина посередині» (Man-in-the-Middle);

— фізична безпека — у разі отримання зловмисником фізичного доступу до пристрою можливе зчитування прошивки, вилучення криптографічних ключів та облікових даних, що потенційно призводить до компрометації всієї мережі IoT [7].

Вирішення цих проблем вимагає комплексного підходу. Необхідна розробка універсальних (підтримують Wi-Fi та BLE), інтелектуальних (здатних до Edge Computing) та безпечних (з апаратною підтримкою шифрування TLS та захистом ключів) мікропроцесорних модулів.

1.2 Аналіз базових апаратних технологій та мікроконтролерних платформ

Сучасний ринок мікропроцесорних засобів для Інтернету речей (IoT) пропонує широкий спектр платформ, вибір між якими залежить від складного балансу між обчислювальною потужністю, енергоспоживанням, набором інтегрованих периферійних пристроїв та кінцевою вартістю. Ці платформи умовно поділяються на дві великі категорії: високопродуктивні одноплатні комп'ютери (SBC) та вузькоспеціалізовані мікроконтролерні платформи (MCU).

Одноплатні комп'ютери, яскравим представником яких є сімейство Raspberry Pi, є повноцінними обчислювальними системами, здатними працювати під управлінням складних операційних систем загального призначення (GPOS), таких як Linux . Вони оснащені потужними багатоядерними мікропроцесорами (наприклад, ARM Cortex-A серії), значним обсягом оперативної пам'яті (часто вимірюється в гігабайтах) та розширеними мультимедійними можливостями. SBC є оптимальним вибором для завдань, що вимагають значних обчислювальних ресурсів, таких як обробка відеопотоків у

реальному часі, виконання алгоритмів машинного навчання на периферії (Edge AI) або функціонування в якості повноцінних мережевих серверів. Проте, їхні суттєві недоліки в контексті універсальних IoT-вузлів є фундаментальними. По-перше, це високе енергоспоживання (вимірюється у ватах), що унеможливлює тривалу автономну роботу від акумуляторних батарей. По-друге, операційні системи загального призначення не є системами реального часу; вони мають недетермінований час відгуку через планувальник завдань, що є неприйнятним для багатьох IoT-застосувань, які вимагають миттєвої та гарантованої реакції на апаратні переривання від сенсорів або виконавчих механізмів.

Мікроконтролерні платформи (MCU), навпаки, розроблені спеціально для вбудованих систем, де пріоритетами є низьке енергоспоживання (вимірюється в мілі- або мікроамперах у режимах сну), компактність та надійне виконання конкретних завдань у реальному часі. Платформа Arduino значною мірою популяризувала розробку на MCU, надавши спрощене середовище програмування (Arduino IDE) та високорівневий API (абстрактний шар апаратного забезпечення), що значно знизило поріг входження для розробників. Класичні плати Arduino (наприклад, Uno) базуються на 8-бітних ядрах ATmega, однак сучасніші рішення використовують 32-бітні процесори ARM Cortex-M. Незважаючи на простоту прототипування, недоліком більшості плат Arduino є відсутність вбудованих засобів бездротової комунікації, що вимагає використання додаткових модулів (shields) для підключення до Wi-Fi або Bluetooth, збільшуючи габарити та енергоспоживання пристрою [8].

Сімейство мікроконтролерів STM32 від STMicroelectronics, що базується на 32-бітних ядрах ARM Cortex-M (від M0 до M7), є де-факто промисловим стандартом для складних та надійних вбудованих систем. Вони пропонують виняткову продуктивність, гнучку систему тактування, чудову енергоефективність та надзвичайно багатий набір периферійних пристроїв (АЦП, ЦАП, таймери, CAN, I2C, SPI). Існують спеціалізовані серії, наприклад STM32WB, які інтегрують підтримку Bluetooth 5 та Zigbee. Однак, для реалізації Wi-Fi зв'язку в більшості випадків потрібен зовнішній модуль, що ускладнює та

здорожчує кінцевий пристрій. Крім того, розробка під STM32 вимагає глибшого розуміння архітектури мікроконтролера та роботи з інструментами (наприклад, STM32CubeMX), що підвищує складність проекту.

Окреме місце в IoT-сегменті займають рішення від компанії Espressif Systems, які з самого початку проектувалися як високоінтегровані системи-на-кристалі (SoC) з вбудованим Wi-Fi. Платформа ESP8266 стала революційною завдяки своїй низькій вартості та наявності 32-бітного ядра Tensilica L106 з повним стеком TCP/IP та підтримкою Wi-Fi. Вона добре підходить для простих IoT-проектів, проте її одноядерна архітектура є фундаментальним недоліком. Одне ядро змушене одночасно обробляти ресурсоємний стек протоколів Wi-Fi та виконувати прикладний код користувача. Це призводить до того, що стек Wi-Fi періодично перериває (preempts) виконання коду програми, викликаючи затримки (jitter) та роблячи неможливою стабільну роботу з протоколами, чутливими до часу, або точне керування виконавчими механізмами [9].

Наступним поколінням стала платформа ESP32, яка була розроблена для усунення ключових недоліків попередника. Вона базується на потужному двоядерному 32-бітному процесорі Xtensa LX6/LX7 (з тактовою частотою до 240 МГц), що дозволяє виконувати паралельні обчислення та реалізовувати справжню багатозадачність. Архітектура ESP32 використовує асиметричну багатопроцесорну обробку (AMP): одне ядро (PRO_CPU, Protocol CPU) виділяється виключно для управління стеками бездротових протоколів (Wi-Fi та Bluetooth 4.2/5.0 LE), тоді як друге ядро (APP_CPU, Application CPU) повністю віддається під логіку користувача. Це кардинально підвищує стабільність, швидкодію системи та дозволяє повноцінно використовувати операційні системи реального часу, такі як FreeRTOS, без компромісів у продуктивності [8].

Платформа Arduino Nano ESP32, яка є предметом даного дослідження, використовує одну з найсучасніших версій цього сімейства — ESP32-S3. Цей чіп оснащений двоядерним процесором Xtensa LX7, має розширені інструкції для прискорення завдань ШІ, великий обсяг вбудованої пам'яті (SRAM), 45 програмованих GPIO та, що найважливіше, вдосконалені апаратні засоби

безпеки. Вони включають апаратне прискорення криптографічних алгоритмів (AES, SHA, RSA), а також механізми Secure Boot (запобігає запуску неавторизованого коду) та Flash Encryption (шифрує прошивку на пристрої) [10].

1.3 Огляд існуючих комерційних рішень та аналогів

Для обґрунтування актуальності та визначення ніші розроблюваного універсального модуля необхідно провести детальний аналіз існуючих на ринку комерційних платформ та комплектів для розробки. Ринок IoT-пристроїв є висококонкурентним, проте більшість рішень є вузькоспеціалізованими, фокусуючись або на мінімальній вартості (з обмеженим функціоналом), або на максимальній інтеграції для конкретної задачі (наприклад, HMI або логістика). Універсальні модулі, що поєднують високу продуктивність, гібридний зв'язок, інтегрований набір сенсорів та засоби керування, зустрічаються рідко. При аналізі ми зосередимося на рішеннях, які пропонують "коробковий" досвід для швидкого підключення до хмарних сервісів.

Одним з найбільш близьких за ідеологією аналогів є плати серії AVR-IoT та PIC-IoT від компанії Microchip (рис. 1.3) [11].

Рисунок 1.3 — Налагоджувальна плата AVR-IoT (PIC-IoT) [11].

Ці платформи були одними з перших, хто запропонував комплексне рішення "з коробки" для швидкого підключення до хмарних сервісів, зокрема

Amazon Web Services (AWS). Їхньою ключовою перевагою, що безпосередньо перетинається з вимогами нашої розробки, є інтеграція на платі криптографічного елемента ATECC608A [11]. Це апаратний "сейф", що дозволяє надійно зберігати приватні ключі та сертифікати, забезпечуючи високий рівень безпеки при підключенні до хмари, що є значною перевагою над простими програмними реалізаціями TLS. Також ці плати включають вбудований зарядний пристрій для Li-Po акумуляторів.

Однак, їхня архітектура має фундаментальний недолік: вона є двочіповою та базується на застарілих мікроконтролерах. Платформа використовує 8-бітний (ATmega4808) або 16-бітний (PIC24FJ128GA705) мікроконтролер для логіки користувача та окремий мережевий контролер (ATWINC1510) для забезпечення Wi-Fi зв'язку [12].

Це створює низку проблем:

— низька продуктивність — 8-бітний мікроконтролер ATmega4808, незважаючи на свою сучасну архітектуру, має обмежені обчислювальні ресурси, що не дозволяє ефективно реалізовувати складні алгоритми периферійних обчислень (Edge Computing) та багатозадачну обробку даних, на відміну від 32-бітного двоядерного мікроконтролера ESP32;

— відсутність BLE — мережевий контролер ATWINC1510 забезпечує роботу виключно в мережах Wi-Fi, що унеможливлює реалізацію гібридних схем зв’язку та спрощеного початкового налаштування пристрою з використанням Bluetooth Low Energy;

— енергоефективність — використання двох окремих функціональних мікросхем замість інтегрованої системи-на-кристалі (SoC) призводить до підвищеного енергоспоживання в активному режимі роботи та ускладнює реалізацію глибоких режимів енергозбереження.

Таким чином, рішення від Microchip, хоч і є безпечними, значно поступаються в продуктивності та гнучкості сучасним 32-бітним SoC.

Інший підхід до універсальності демонструє компанія M5Stack, зокрема її

флагманський продукт M5Stack CoreS3 (рис. 1.4). Ця платформа базується на тому ж сучасному чіпі ESP32-S3, що й обраний нами Nano ESP32. Її головна ідея — максимальна інтеграція у готовому до використання корпусі. Вона пропонує надзвичайний рівень "коробкового" рішення: високоякісний 2-дюймовий сенсорний IPS-дисплей, вбудований слот для MicroSD-карти, динамік, мікрофон, 30-піновий роз'єм розширення та вбудований акумулятор. Це робить M5Stack ідеальним рішенням для завдань, що вимагають складної взаємодії з користувачем (HMI, Human-Machine Interface).

Рисунок 1.4 — Зовнішній вигляд платформи M5Stack CoreS3 [13]

Проте, ця платформа є спеціалізованою саме для HMI-додатків. Її архітектура не призначена для використання як універсального сенсорного вузла. По-перше, вона не має вбудованих сенсорів навколишнього середовища (як BME280 або BH1750), а її власний температурний датчик призначений для моніторингу самого чіпа. Розміщення зовнішніх кліматичних сенсорів поруч із корпусом, що нагрівається від потужного дисплея та CPU, призведе до спотворення вимірювань. По-друге, платформа не надає зручних виходів для керування силовими навантаженнями (MOSFET). Вона розрахована на власну екосистему модулів Grove, що обмежує універсальність.

Платформи серії Adafruit Feather є еталоном для швидкого прототипування завдяки стандартизованому форм-фактору (Feather) та наявності роз'єму STEMMA QT [14]. Цей роз'єм дозволяє миттєво підключати десятки різноманітних I2C-сенсорів та модулів без необхідності пайки. Плати на базі ESP32-S3 (як Adafruit Feather ESP32-S3) є прямими конкурентами Nano ESP32 і також включають вбудований контролер заряду Li-Po акумулятора (рис 1.5).

Однак, філософія Adafruit — це "плата для розробки", а не "інтегрований модуль". Як і M5Stack, плата Adafruit Feather сама по собі не містить жодного інтегрованого сенсора, годинника реального часу чи модуля NFC. Вона є лише "мозковим центром", який повністю покладається на зовнішні, хоч і зручні для підключення, модулі. Для створення готового, компактного кінцевого пристрою такий підхід є менш оптимальним. Він призводить до "бутербродної" конструкції, що збільшує загальні габарити, вартість та кількість точок відмови (конектори, дроти) порівняно з рішенням, де ключові компоненти вже інтегровані на одній друкованій платі.

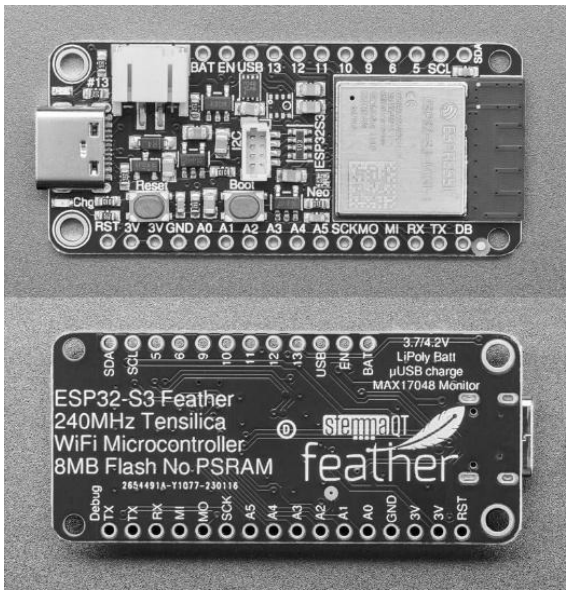


Рисунок 1.5 —Зовнішній вигляд Adafruit Feather ESP32-S3 [14].

Платформи від компанії Particle (раніше Spark Core) є ще одним важливим

гравцем на ринку. Їхній модуль Particle Argon (рис. 1.6) є потужним рішенням, що поєднує мікроконтролер nRF52840 (ARM Cortex-M4 з підтримкою BLE 5.0 та Mesh) та ESP32 (який використовується суто як Wi-Fi співпроцесор). Це забезпечує високу продуктивність та надійний гібридний зв'язок.

Основний недолік цієї платформи — тісна прив'язка до власної хмарної екосистеми (Particle Cloud). Хоча це значно спрощує розробку (надаючи готові OTA, веб-хуки та API), це також створює форму "vendor lock-in" (прив'язка до постачальника) [15].

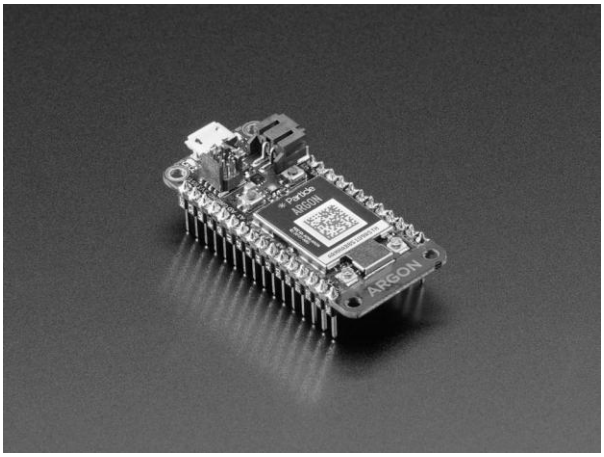


Рисунок 1.6 — Зовнішній вигляд платформи Particle Argon [15]

Розроблюваний нами модуль, навпаки, орієнтований на відкриті стандарти (чистий MQTT-TLS), що дозволяє підключати його до будь-якого брокера, будь то публічний хмарний сервіс (AWS, Azure) або приватний локальний сервер (наприклад, Mosquitto), що забезпечує значно вищий рівень універсальності.

Існують комплексні комерційні набори, такі як IoT Mill (рис. 1.7).
Ця платформа призначена для швидкого розгортання промислових IoT-рішень і є значно складнішою за попередні аналоги [16]. Її головна перевага полягає у підтримці мульти-радіо доступу: окрім Wi-Fi та Bluetooth, вона підтримує стільникові мережі LTE-M/NB-IoT та має вбудований GPS. Це робить її ідеальною для завдань моніторингу рухомих об'єктів (трекінг активів, логістика). Однак, для стаціонарних IoT-завдань (розумний дім, моніторинг клімату) такий функціонал є надлишковим та значно збільшує вартість і

енергоспоживання. Крім того, ця платформа має значні габарити, орієнтована на хмарні сервіси розробника і не має вбудованого графічного дисплея чи гнучких виходів керування, як MOSFET.

Рисунок 1.7 — Зовнішній вигляд платформи IoT Mill для промислових застосувань [16]

Проведений аналіз показує, що на ринку відсутнє рішення, яке б поєднувало в собі всі необхідні для нашого завдання характеристики. Існуючі аналоги є або архітектурно застарілими та малопродуктивними (Microchip), або вузькоспеціалізованими (M5Stack для HMI, IoT Mill для логістики), або є просто платами для розробки, що вимагають значної кількості зовнішніх модулів (Adafruit Feather), або є закритими екосистемами (Particle).

Це підтверджує наявність ринкової ніші та актуальність розробки власного універсального мікропроцесорного модуля, який інтегрує на одній компактній платформі:

— потужний мікроконтролер;

— повний набір кліматичних сенсорів;

— компоненти для автономної роботи, такі як наприклад реального часу, контролер зарядки акумулятора, індикатор;

— локальне сховище (MicroSD);

— різноманітні інтерфейси взаємодії.

1.4 Постановка завдань дослідження

Проведений у попередніх підрозділах аналіз сучасного стану Інтернету речей, огляд мікроконтролерних платформ та дослідження існуючих ринкових аналогів дозволяє зробити висновок про наявність чітко окресленої науково-технічної проблеми. Було встановлено, що, незважаючи на велику кількість IoT-пристроїв, на ринку відсутнє універсальне рішення, яке б одночасно задовольняло суперечливий набір вимог.

Сучасні аналоги часто є або архітектурно застарілими, як двочіпові рішення Microchip на 8-бітних МК [11], або вузькоспеціалізованими, як M5Stack для HMI [13] чи IoT Mill для логістики [16]. Таким чином, ринок потребує гнучкого рішення, що поєднує високу продуктивність для граничних обчислень (Edge Computing) та багатозадачності в реальному часі, що неможливо на одноядерних платформах. Окрім цього, критично необхідною є підтримка гібридного зв'язку (Wi-Fi та Bluetooth LE), надійні апаратні засоби безпеки (для TLS/MQTTS) та гнучкі режими глибокого сну для тривалої автономної роботи.

Вирішенням цієї проблеми є розробка власного універсального мікропроцесорного модуля на базі сучасної двоядерної платформи Nano ESP32 (ESP32-S3), яка апаратно задовольняє вимоги до продуктивності, гібридного зв'язку та безпеки. Однак, сам по собі вибір платформи не вирішує завдання енергоефективності та надійності програмного забезпечення.

Тому, на основі проведеного аналізу, метою даної магістерської роботи є розробка та експериментальне дослідження універсального мікропроцесорного модуля на базі Nano ESP32 для підвищення його енергоефективності та функціональної надійності в IoT-застосуваннях.

Для досягнення поставленої мети необхідно послідовно вирішити низку науково-технічних завдань.

По-перше, необхідно провести глибоке теоретичне дослідження обраних технологій, детально проаналізувавши архітектурні моделі обміну даними

(M2M, Cloud, Fog) та дослідити особливості реалізації протоколів зв’язку та безпеки (Wi-Fi, BLE, MQTT, TLS) в контексті вбудованих систем.

По-друге, слід розробити повну апаратну архітектуру універсального модуля, що інтегрує підсистему моніторингу (BME280, BH1750), підсистему автономної роботи (RTC DS3231, MicroSD) та підсистему керування (MOSFET-ключі). Цей етап включає розробку детальної функціональної схеми та розрахунок автономної системи живлення.

По-третє, потрібно розробити багатозадачне програмне забезпечення модуля на базі операційної системи реального часу FreeRTOS, що реалізує паралельне виконання завдань збору даних, локального керування та безпечної гібридної комунікації.

По-четверте, центральним завданням є розробка та програмна реалізація адаптивного алгоритму керування живленням, що інтелектуально керує режимами сну (Deep Sleep, Light Sleep) мікроконтролера залежно від зовнішніх подій та завдань моніторингу.

По-п'яте, необхідно провести експериментальне дослідження розробленого модуля для вимірювання та його реальних показників енергоспоживання та швидкодії, з подальшим порівняльним аналізом ефективності з аналогічними рішеннями.

2 ТЕОРЕТИЧНІ ОСНОВИ АРХІТЕКТУР ТА ПРОТОКОЛІВ IOT СИСТЕМ

2.1 Аналіз архітектур мікропроцесорних систем та методів обміну даними

Ефективність та надійність будь-якого сучасного IoT-рішення визначається не стільки обчислювальною потужністю окремого кінцевого пристрою, скільки архітектурою його взаємодії в межах складної, багаторівневої інфраструктури обміну даними. Універсальний мікропроцесорний засіб (МПЗ), що є предметом розробки, повинен функціонувати як надійний інтелектуальний вузол, здатний до автономної роботи та ефективної взаємодії з вищим рівнем обчислень. Фундаментальною концепцією, що лежить в основі цієї взаємодії, є технологія міжмашинної взаємодії (Machine-to-Machine, M2M). M2M забезпечує автоматизований збір, передачу та обмін даними між апаратними пристроями без необхідності постійного втручання людини [6]. Розроблюваний модуль, який зчитує показники сенсорів та керує виконавчими механізмами, виступає саме таким M2M-пристроєм на «нижньому» рівні IoT-ієрархії.

Централізована архітектура: Парадигма Хмарних обчислень (Cloud Computing) для обробки та довгострокового зберігання петабайтів даних, що генеруються мільярдами IoT-пристроїв, традиційно застосовується парадигма хмарних обчислень (Cloud Computing) [4]. Хмарні платформи (такі як AWS IoT, Google Cloud Platform, Microsoft Azure) надають практично необмежені, еластичні ресурси для масштабування сховищ, проведення глибокого аналізу (Big Data analytics) та візуалізації телеметрії. У такій архітектурі кінцевий МПЗ (сенсорний вузол) збирає інформацію (наприклад, стан навколишнього середовища) і періодично відправляє сирий (raw) або мінімально оброблений масив даних до віддаленого централізованого сервера, де і відбувається прийняття всіх ключових рішень.

Незважаючи на потужність хмарних рішень, цей централізований підхід має два суттєві недоліки, критичні для багатьох застосунків. По-перше, виникає значна мережева затримка (latency), оскільки дані мають пройти довгий шлях до

центру обробки та повернутися назад, що може займати сотні мілісекунд або навіть секунди. Для додатків, які вимагають реакції в реальному часі (наприклад, контроль промислових процесів, системи безпеки, автономне водіння), така затримка є неприйнятною. По-друге, постійна передача великих обсягів сирих даних навантажує канали зв'язку та вимагає високої пропускної здатності, а також призводить до значного енергоспоживання кінцевих пристроїв.

Для усунення проблеми затримок та зниження навантаження на магістральні канали зв’язку була розроблена концепція туманних обчислень (Fog Computing) [5]. Архітектура Fog Computing є розподіленою моделлю, яка розширює хмарні сервіси та обчислювальні ресурси ближче до джерела даних — на «край» мережі (Edge). «Туманний» шар, розташований між кінцевими пристроями та глобальною хмарою, виконує роль проміжного брокера та процесора.

Архітектура ділиться на три рівні:

— cloud (хмара) — це централізоване сховище та глобальна аналітика;

— fog (туман) — це локальні сервери (шлюзи), що обробляють дані від групи пристроїв;

— edge (край) — це кінцеві пристрої (сенсори, виконавчі механізми), які генерують дані.

Основна мета Fog-вузла — здійснювати попередню обробку, фільтрацію, агрегацію та стиснення даних безпосередньо в місці їх генерації. Це дозволяє приймати миттєві, локальні рішення (Edge Decision-Making) на основі критичних даних, не чекаючи відповіді від віддаленого сервера. Наприклад, якщо температура перевищила встановлену межу, Fog-вузол миттєво активує систему охолодження. У хмару ж після локальної обробки відправляються лише узагальнені звіти, історія подій або критичні аномалії, що значно зменшує необхідний трафік. Таким чином, Fog Computing не замінює, а доповнює Cloud Computing, забезпечуючи низьку затримку для оперативного контролю та

високу потужність для довгострокового аналізу відповідно [5]. Концепція Fog Computing як проміжної ланки між пристроями та хмарою показана на рисунку 2.1.

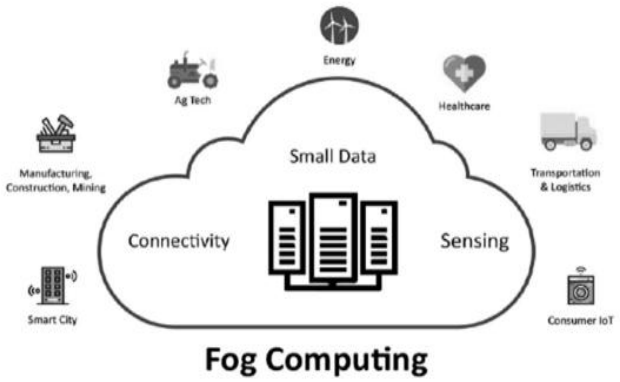


Рисунок 2.1 — Схематична модель архітектури туманних обчислень (Fog Computing) [5]

Саме тут розкриваються ключові архітектурні переваги обраної платформи Nano ESP32 (на базі ESP32-S3), яка ідеально підходить для ролі інтелектуального Fog/Edge вузла. Реалізація локальної обробки та прийняття рішень вимагає детермінізму — гарантованого часу відгуку, незалежно від стану мережі. Це неможливо забезпечити на одноядерних системах (наприклад, ESP8266), де одне ядро повинне одночасно обробляти ресурсоємний стек протоколів Wi-Fi (що вимагає великої кількості переривань) та виконувати логіку прикладного коду (опитування сенсорів, управління виходами).Конфлікти за ресурси призводять до збоїв, затримок та нестабільності.

Платформа ESP32-S3, завдяки двоядерній архітектурі та інтеграції операційної системи реального часу (FreeRTOS), вирішує цю проблему шляхом застосування принципу симетричної багатопроцесорної обробки (SMP), адаптованого Espressif. FreeRTOS дозволяє чітко розділити системні та прикладні завдання на два незалежні ядра.

Ядро 0 (PRO_CPU) зазвичай використовується для критичних системних завдань. На нього «закріплюються» (pinning) такі функції, як управління стеком Wi-Fi, обробка криптографії (TLS/SSL) та низькорівнева обробка мережевих пакетів. Це забезпечує стабільну роботу комунікаційного каналу, запобігаючи його блокуванню.

Ядро 1 (APP_CPU) виділяється виключно для виконання логіки користувача. На цьому ядрі запускаються завдання з опитування сенсорів (BME280, BH1750), реалізація алгоритмів аналізу даних, прийняття локальних рішень та безпосереднє керування виконавчими механізмами (MOSFET-виходами) [8].

Таке розділення завдань гарантує, що ресурсоємна мережева активність ніколи не зможе заблокувати виконання критичного коду, який відповідає за збір даних та керування системою. Цей підхід перетворює Nano ESP32 з простого "передавача даних" у хмару на високоінтелектуальний, надійний та детермінований Edge-вузол, що відповідає вимогам архітектури туманних обчислень.

2.2 Дослідження протоколів зв’язку та безпеки.

Вибір апаратної платформи є лише першим кроком у проектуванні IoT-системи. Не менш важливим є вибір стеку протоколів зв'язку, оскільки він безпосередньо визначає енергоефективність, швидкодію, надійність та безпеку всього рішення. Для реалізації по-справжньому універсального модуля недостатньо мати лише один канал зв'язку. Система повинна підтримувати гібридний набір протоколів для вирішення різних класів завдань: від високошвидкісної передачі даних до Інтернету до локальної конфігурації з ультранизьким енергоспоживанням. Обрана платформа Nano ESP32 (на базі ESP32-S3) є ідеальним кандидатом для такої гібридної системи, оскільки її радіомодуль апаратно підтримує два ключових бездротових стандарти: Wi-Fi та

Bluetooth LE. Демонстрація інтегрованих блоків бездротового зв'язку та криптографії, наведена на рисунку 2.2.

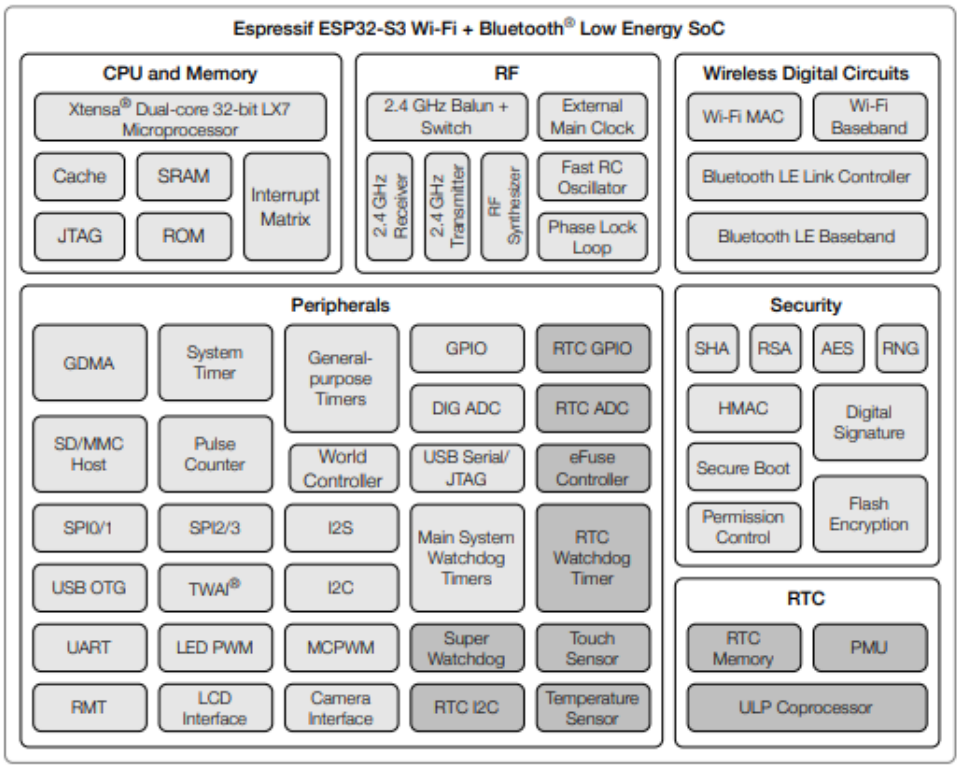


Рисунок 2.2 — Блок-схема апаратних засобів ESP32-S3 для реалізації гібридного зв'язку (Wi-Fi та Bluetooth LE) [10]

Wi-Fi (IEEE 802.11 b/g/n) є фундаментальною технологією для IoT-пристроїв, що потребують прямого підключення до Інтернету через IP-мережу. Його головна перевага — висока пропускна здатність (десятки Мбіт/с) та відносно великий радіус дії (десятки метрів у приміщенні), що дозволяє передавати великі обсяги даних, наприклад, пакети оновлення прошивки "по повітрю" (OTA Updates) [17]. Однак, Wi-Fi є надзвичайно енергоємним протоколом. Процес сканування мереж, асоціації з точкою доступу (AP), виконання DHCP-запиту для отримання IP-адреси та підтримання активного TCP/IP-з'єднання вимагає значних витрат енергії. Пікове споживання струму під час передачі даних (Tx) може сягати сотень міліампер. Для пристроїв, що розробляються з вимогою тривалої автономної роботи від акумулятора, постійно

активний Wi-Fi є неприпустимою розкішшю. Тому його використання має бути оптимізовано: модуль повинен активувати Wi-Fi лише на короткі проміжки часу для відправки накопичених даних, а решту часу (99% і більше) перебувати в режимі глибокого сну (Deep Sleep). Саме реалізація цього адаптивного циклічного режиму роботи є одним з ключових завдань магістерської роботи.

Bluetooth Low Energy (BLE), на відміну від "класичного" Bluetooth, був розроблений спеціально для IoT-застосувань з пріоритетом на ультранизьке енергоспоживання. BLE працює в тому ж ISM-діапазоні 2.4 ГГц, але використовує інший механізм модуляції (GFSK), швидке встановлення з'єднання та значно коротші пакети даних [18]. Його ключова перевага — здатність підтримувати зв'язок або оголошувати про свою присутність (advertising) зі споживанням струму на рівні мікроампер. В архітектурі універсального IoT-модуля BLE відіграє дві критичні ролі, які Wi-Fi не може ефективно виконати. По-перше, це початкове налаштування (Provisioning). Коли користувач вперше вмикає пристрій, модуль не знає облікових даних локальної мережі Wi-Fi. Замість складних та небезпечних методів, модуль активує BLE. Користувач за допомогою мобільного додатку на смартфоні підключається до модуля, безпечно передає йому облікові дані домашньої мережі Wi-Fi (SSID та пароль), після чого BLE може бути вимкнено. По-друге, це локальний моніторинг та керування. BLE дозволяє взаємодіяти з пристроєм безпосередньо, без необхідності підключення до Інтернету чи хмарного сервера, що критично важливо для додатків, де потрібна низька затримка (наприклад, керування MOSFET-виходами) або у випадку, коли основний канал зв'язку (Wi-Fi) тимчасово недоступний. Саме гібридна підтримка Wi-Fi та BLE у чіпі ESP32-S3 є його визначальною перевагою [10].

На прикладному рівні HTTP (Hypertext Transfer Protocol) є основою класичного вебу. Використання HTTP (або HTTPS) є простим для взаємодії з RESTful API веб-сервісів. Однак для IoT він має суттєві недоліки, головний з яких — висока надлишковість (overhead). Кожен HTTP-запит містить великі текстові заголовки (headers), які часто за обсягом у десятки разів перевищують

корисні дані. Крім того, синхронна модель "запит-відповідь" є блокуючою та вимагає встановлення нового TCP-з'єднання для кожного повідомлення, що вкрай неефективно для частої відправки телеметрії.

MQTT (Message Queuing Telemetry Transport) є де-факто стандартом для IoT-комунікацій. Це надзвичайно легкий бінарний протокол, що працює за моделлю "видавець-підписник" (Publish/Subscribe) поверх TCP/IP. Замість прямого з'єднання, пристрої (видавці) відправляють повідомлення у певні "теми" (topics) на центральний сервер (Брокер). Інші пристрої (підписники) підписуються на ці теми та миттєво отримують повідомлення. Ця архітектура декаплінгу (decoupling) означає, що видавець і підписник не знають про існування один одного, що забезпечує надзвичайну гнучкість системи. Переваги MQTT (схема взаємодії наведена на рис. 2.3) для IoT-модуля є фундаментальними: легкість (мінімальний розмір заголовка від 2 байт), асинхронність (пристрій відправляє дані і не чекає відповіді, що ідеально для енергозбереження), надійність (підтримка трьох рівнів QoS) та ефективність (підтримка тривалих TCP-сесій) [19].

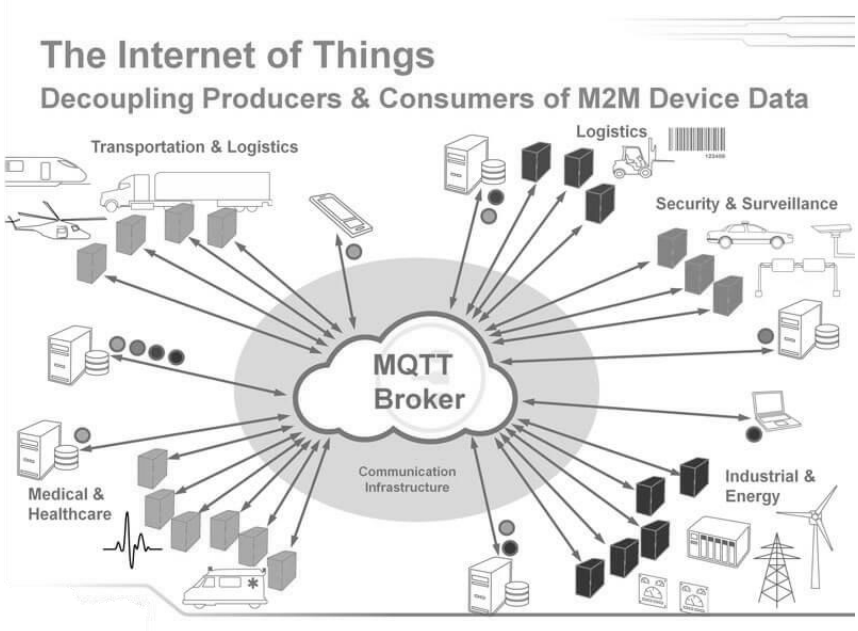


Рисунок 2.3 — Архітектура взаємодії пристроїв через MQTT Broker [27]

Безпека є однією з найголовніших проблем сучасного IoT. Передача даних

телеметрії у відкритому вигляді створює неприйнятні ризики. Для захисту каналу зв'язку використовується TLS (Transport Layer Security). TLS забезпечує три стовпи безпеки: конфіденційність (шифрування, наприклад, AES), автентифікацію (перевірка сторін за допомогою сертифікатів X.509) та цілісність (захист від модифікації через MAC). Обраний протокол MQTT сам по собі нешифрований (порт 1883), тому для його захисту використовується MQTT-TLS (MQTTS), що є тим самим протоколом MQTT, "загорнутим" у захищений тунель TLS (порт 8883) [7].

Проблема полягає в тому, що реалізація TLS є надзвичайно ресурсоємною задачею для мікроконтролерів. Найскладнішою частиною є процес TLS-рукостискання (handshake) — асиметрична криптографічна операція, яка на платформах без апаратної підтримки (як ESP8266) може займати кілька секунд. Для автономного пристрою, який повинен миттєво відправити дані і заснути, такі затримки є катастрофічними. Платформа ESP32-S3, обрана для Nano ESP32, вирішує цю проблему шляхом інтеграції апаратних криптографічних прискорювачів. Вона має вбудовані модулі для виконання операцій AES, SHA, RSA та ECC. Це дозволяє делегувати (offloads) найскладніші обчислення рукостискання на апаратні модулі, звільняючи CPU та скорочуючи час встановлення TLS-з'єднання з секунд до мілісекунд. Завдяки цьому, реалізація захищеного MQTTS-з'єднання стає не лише можливою, але й енергоефективною.

2.3 Теоретичні основи локальних інтерфейсів та протоколів (I2C, SPI)

При проектуванні універсального мікропроцесорного модуля, вибір локальних інтерфейсів (шин) для зв'язку між центральним мікроконтролером (Nano ESP32) та периферійними пристроями (сенсорами, пам'яттю, дисплеями) є не менш важливим, ніж вибір самого МК. Неправильно обрана шина може стати "вузьким місцем" системи, обмежуючи швидкість передачі даних, або невиправдано займати велику кількість цінних виводів (GPIO) мікроконтролера.

У сучасній схемотехніці вбудованих систем домінують два послідовні протоколи: I2C та SPI [20].

Інтерфейс I2C (Inter-Integrated Circuit), розроблений компанією Philips (зараз NXP) у 1980-х роках, є де-факто стандартом для підключення повільної та середньошвидкісної периферії. Це синхронна, багатоточкова (multi-master, multi-slave) шина, яка для роботи вимагає лише двох ліній:

— SDA (Serial Data Line) — це двонаправлена лінія для передачі даних;

— SCL (Serial Clock Line) — це лінія для передачі тактових імпульсів, яку генерує ведучий пристрій (Master), у нашому випадку — Nano ESP32.

Ключовою особливістю I2C є її архітектура з "відкритим колектором" (open-drain). Це означає, що пристрої можуть лише "притягувати" лінію до низького рівня (GND), але не можуть генерувати високий рівень (+3.3 V). Для забезпечення високого рівня та, власне, функціонування шини, обидві лінії (SDA та SCL) (рис. 2.4) обов'язково вимагають зовнішніх підтягуючих резисторів (pull-up resistors), що підключаються до лінії живлення +3.3 V [21].

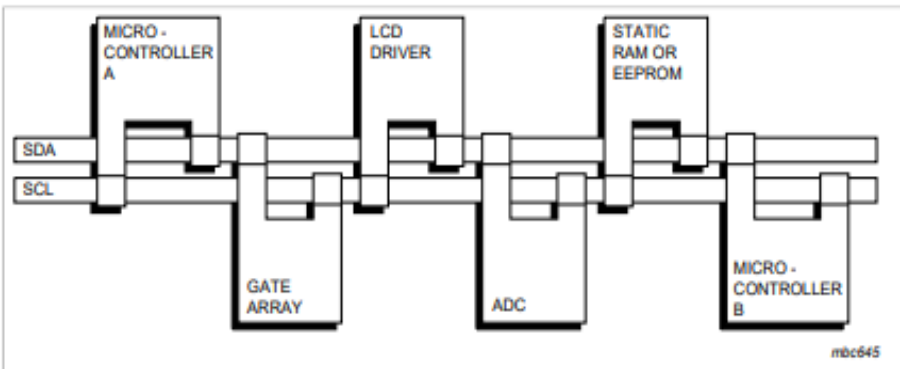


Рисунок 2.4 — Топологія шини I2C [20]

Адресація пристроїв на шині I2C є програмною. Кожен пристрій-ведений (slave) має унікальну 7-бітну (або 10-бітну) адресу, яка або жорстко задана виробником (наприклад, 0x76 для BME280), або налаштовується апаратно (наприклад, пінами A0/A1/A2 на MCP23017). Це дозволяє підключити десятки пристроїв (теоретично до 127) до тих самих двох пінів мікроконтролера, що є головною перевагою I2C для економії виводів. Недоліком є відносно невисока

швидкість (зазвичай 100 кГц "Standard-mode" або 400 кГц "Fast-mode") та напівдуплексний режим (передача та прийом не можуть відбуватися одночасно) [22]. Це робить I2C ідеальним для наших сенсорів (BME280, BH1750) та RTC (DS3231), які передають малі обсяги даних.

Інтерфейс SPI (Serial Peripheral Interface), розроблений компанією Motorola, є іншим підходом до послідовної передачі даних. Це також синхронна шина з архітектурою "Master-Slave", але вона призначена для високошвидкісної передачі даних. На відміну від I2C, SPI використовує окремі лінії для передачі та прийому, що забезпечує повнодуплексний зв'язок (одночасна передача та прийом). Класична SPI вимагає щонайменше чотирьох ліній [21]:

— MISO (Master In, Slave Out) — це лінія даних від веденого до ведучого;

— MOSI (Master Out, Slave In) — це лінія даних від ведучого до веденого;

— SCK (Serial Clock) — це лінія тактових імпульсів від ведучого;

— CS (Chip Select) або SS (Slave Select) — це індивідуальна лінія для кожного веденого пристрою (рис. 2.5).

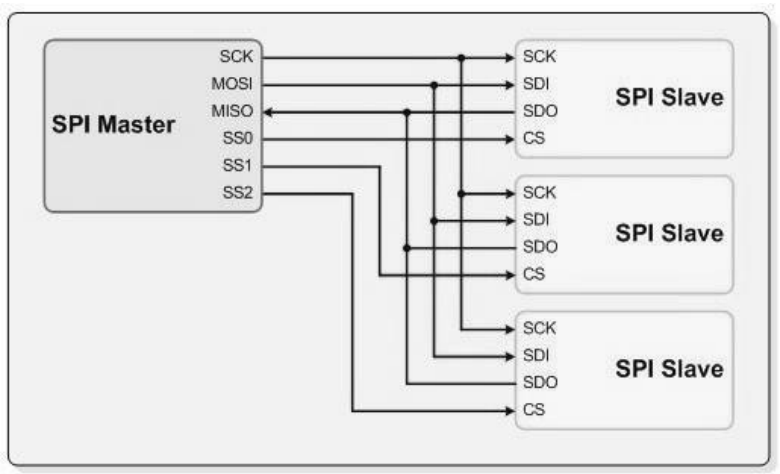


Рисунок 2.5 — Схема підключення декількох пристроїв за інтерфейсом SPI [28]

Адресація в SPI є апаратною. Ведучий (ESP32) активує ведений пристрій, подаючи низький логічний рівень на його персональний пін CS. Інші пристрої, у яких CS залишається у високому стані, ігнорують активність на шині.

Це є головним недоліком SPI: для підключення N пристроїв потрібно N+3 виводів мікроконтролера. Однак перевага у швидкості є колосальною — стандартні швидкості SPI сягають десятків МГц (наприклад, 20-80 МГц).

Саме тому SPI є єдиним прийнятним вибором для таких завдань, як запис на MicroSD-карту (де потрібна висока швидкість для логування даних) або взаємодія з NFC-контролером (PN532), де обмін даними в режимі емуляції карти також вимагає швидкого відгуку. Використання I2C для цих завдань призвело б до неприйнятних затримок.

2.4 Принципи побудови багатозадачних вбудованих систем та методи енергозбереження

При проектуванні програмного забезпечення для сучасного IoT-пристрою, який повинен одночасно виконувати декілька асинхронних завдань (наприклад, опитувати сенсори, оновлювати дисплей та підтримувати Wi-Fi з'єднання), традиційний підхід програмування у вигляді нескінченного циклу (відомий як "superloop") стає неефективним. Будь-яка тривала операція, як-от спроба підключення до мережі або запис на SD-карту, призводить до блокування всієї системи, унеможливлюючи виконання інших, можливо, більш критичних завдань у реальному часі.

Для вирішення цієї проблеми у професійних вбудованих системах застосовуються операційні системи реального часу (ОСРЧ, або RTOS). На відміну від операційних систем загального призначення (як Linux), RTOS гарантує детермінізм — здатність системи реагувати на події та виконувати завдання у чітко визначені, передбачувані проміжки часу [23]. Платформа ESP32-S3, обрана для нашої розробки, використовує FreeRTOS як невід'ємну частину свого основного програмного каркасу (ESP-IDF) [24].

Основою архітектури FreeRTOS є концепція завдань (Tasks). Кожне завдання — це незалежний потік виконання (по суті, окрема міні-програма) зі своїм власним стеком та пріоритетом. Ключову роль відіграє планувальник

(Scheduler) FreeRTOS. Це компонент ядра, який постійно вирішує, яке завдання має виконуватися в даний момент часу. Планувальник FreeRTOS є витісняючим (preemptive): якщо з'являється завдання з вищим пріоритетом, яке готове до виконання, планувальник негайно призупиняє (swaps out) поточне завдання з нижчим пріоритетом і передає керування процесору більш важливому завданню [23]. Життєвий цикл завдання в системі та логіка переходів між станами під управлінням планувальника FreeRTOS наведені на рисунку 2.6

Для безпечної взаємодії між цими паралельними завданнями FreeRTOS надає примітиви синхронізації. Найважливішими для нашого проекту є м'ютекси (Mutexes), які є механізмами блокування, що гарантують ексклюзивний доступ до спільного ресурсу. Наприклад, щоб уникнути конфліктів, коли два завдання одночасно намагаються отримати доступ до шини I2C (як описано в 2.3), вони повинні спочатку "взяти" м'ютекс. Іншим важливим примітивом є черги (Queues) — потокобезпечні буфери FIFO, що є основним механізмом передачі даних між завданнями.

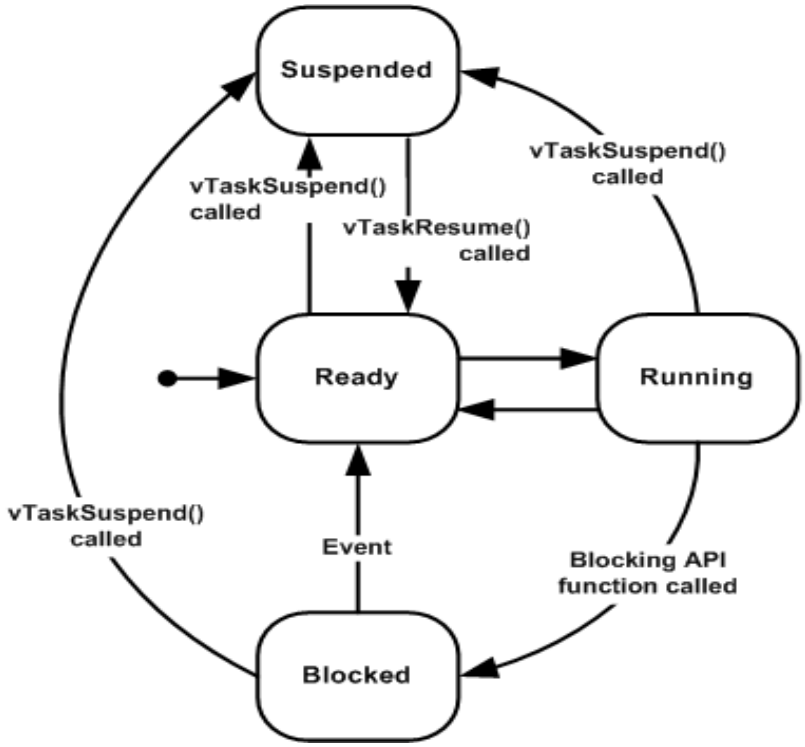


Рисунок 2.6 — Діаграма станів задач та переходів у ядрі FreeRTOS [31]

Як було зазначено раніше, двоядерна архітектура ESP32-S3 дозволяє реалізувати цю модель найбільш ефективно, "прив'язуючи" (pinning) мережеві завдання до одного ядра (PRO_CPU), а завдання користувача — до іншого (APP_CPU), що забезпечує справжній паралелізм та високу надійність системи [24].

Для IoT-пристроїв, що працюють від акумулятора, мінімізація енергоспоживання є пріоритетним завданням. Сучасні мікроконтролери, включаючи ESP32-S3, пропонують для цього розширені апаратні механізми управління живленням. Вся стратегія енергозбереження будується на тому, щоб тримати систему в Активному режимі (Active Mode) якомога менше часу, а решту часу проводити в одному з режимів сну [25].

Активний режим (Active Mode) характеризується споживанням десятків та сотень міліампер (від 80 мА до 250 мА). В цьому режимі обидва ядра CPU працюють на повній частоті (до 240 МГц), радіомодулі (Wi-Fi, BLE) активні, а вся периферія тактується. Цей режим необхідний для передачі даних, але має бути максимально коротким.

Режим глибокого сну (Deep Sleep Mode) є основним станом очікування для автономного модуля. В цьому режимі споживання падає до мікроампер (зазвичай ~10-30 мкА). Таке низьке споживання досягається шляхом повного відключення живлення основних ядер CPU, більшості периферії та очищення вмісту оперативної пам'яті (SRAM). У цьому стані активними залишаються лише годинник реального часу (RTC) та, опціонально, співпроцесор ULP (Ultra-Low-Power). Пробудження з режиму Deep Sleep апаратно еквівалентне повному перезавантаженню (reset) системи. Система може бути "розбуджена" декількома джерелами: спрацюванням таймера RTC або зміною рівня сигналу на одному зі спеціальних RTC GPIO пінів [8, 24].

Режим легкого сну (Light Sleep Mode) є проміжним станом. Його споживання вище, ніж у Deep Sleep (зазвичай ~1-2 мА), але він має ключову перевагу: оперативна пам'ять (SRAM) та периферія зберігають свій стан та живлення, а ядра CPU лише призупиняються. Пробудження з цього режиму

відбувається значно швидше (за мікросекунди) і не вимагає перезавантаження системи — програма продовжує виконання з того місця, де була зупинена. Цей режим ідеально підходить для завдань, що вимагають частого, але короткого пробудження, наприклад, для підтримки активного BLE-з'єднання, де модуль повинен швидко реагувати на запити від смартфона [26, 25].

Таким чином, теоретична база FreeRTOS дозволяє створити надійне багатозадачне ПЗ, а теоретичні основи режимів сну ESP32 дають інструментарій для розробки складних адаптивних алгоритмів енергозбереження.

3 РОЗРОБКА АПАРАТНО-ПРОГРАМНИХ ЗАСОБІВ МОДУЛЯ

3.1 Розроблення структурної схеми універсального модуля

Розробка апаратної архітектури є ключовим етапом проектування, що визначає загальну компоновку системи та принципи взаємодії її складових. Структурна схема є високорівневим представленням, яке ілюструє основні функціональні вузли пристрою та інформаційні потоки між ними, не вдаючись у деталі конкретних електричних з'єднань [27]. Архітектура універсального мікропроцесорного модуля проектувалася з урахуванням вимог модульності, енергоефективності та функціональної повноти.

Розроблюваний засіб будемо будувати на основі мікроконтролерної платформи — готового рішення, що на одній друкованій платі містить мікроконтролер, елементи живлення, інтерфейсні модулі. Завдяки цьому мікроконтролерна платформа є зручною базою для розробки та тестування різноманітних вбудованих систем. Для вирішення завдань даної магістерської роботи підійде платформа, в якій реалізована підтримка бездротових інтерфейсів Wi-Fi та Bluetooth, наприклад одна з тих, що були розглянуті у підрозділі 1.2.

Розроблюваний засіб має бути універсальним, тобто забезпечувати легку інтеграцію у різні застосування IoT. Сьогодні найбільшу популярність отримали засоби IoT систем «Розумний будинок» або «Розумний офіс». Це пов'язано з тим, що з одного боку зазначені системи дозволяють підвищити комфорт та енергоефективність, а з іншого — не вимагають будь-яких спеціальних знань та спеціального обладнання.

В системах, що у тому чи іншому ступені належать до категорії систем «Розумний будинок», пристрої IoT забезпечують у реальному часі управління клімат-контролем, освітленням, опаленням та побутовими пристроями. Виходячи з цього розроблюваний апаратно-програмний засіб повинен містити у своєму складі сенсори температури, вологості, освітленості, руху, мати засоби комутації електричних кіл, вести відлік реального часу. Наявність зазначених

компонентів використовувати розроблюваний модуль як готове універсальне рішення для вирішення задач контролю температури, волості, освітлення, управління різними засобами у реальному часі для управління значеннями цих контрольованих параметрів.

З врахуванням усього цього було розроблено структурну схему універсальний мікропроцесорного модуля для IoT застосувань, що наведена у додатку В. В основі модуля лежить мікроконтролерний блок з вбудованою підтримкою Wi-Fi та Bluetooth з’єднань. Це головний функціональний вузол, що забезпечить виконання усіх основних функцій кінцевого продукту IoT, керуючи всіма іншими периферійними блоками та реалізуючи всю програмну логіку, включаючи виконання завдань FreeRTOS, обробку даних та управління бездротовими інтерфейсами. Всі інші компоненти системи логічно групуються у функціональні підсистеми, що підключаються до мікроконтролерного блоку через стандартизовані цифрові шини.

Підсистема збору даних (блок сенсорів) об'єднує набір сенсорів, призначених для моніторингу параметрів навколишнього середовища. До неї включені сенсори температури, вологості, атмосферного тиску для контролю та управління параметрами мікроклімату, а також сенсор освітленості для оцінки рівня світлового потоку в задачах автоматизації освітлення.

Для підключення сенсорних вузлів обрано послідовну шину I2C. Такий вибір обґрунтований можливістю адресації багатьох пристроїв по двох лініях даних, що дозволяє економити ресурси мікроконтролера та спрощує трасування плати. Швидкості шини I2C (стандартно від 100 до 400 кбіт/с) більш ніж достатньо для періодичного опитування повільних кліматичних сенсорів.

Наступною є підсистема хронометрії та зберігання даних, що забезпечує можливість реалізації функціональних завдань у реальному часі та локального зберігання даних, що є критично важливо для забезпечення автономності та цілісності даних. Система включає годинник реального часу (RTC) та блок енергонезалежної пам’яті. Годинник реального часу є автономним модулем з власним резервним живленням, що відлік часових міток незалежно від

наявності живлення чи підключення до Інтернету. Блок енергонезалежної пам’яті забезпечує буферизацію даних при відсутності з'єднання з мережею. Для забезпечення високої швидкості запису масивів даних цей блок підключається через високошвидкісний інтерфейс SPI, що надає більшою швидкістю передачі даних, ніж шина I2C.

Підсистема керування включає компоненти керування електричним навантаженнями. До неї входить блок силової комутації, реалізований на базі напівпровідникових ключів, що дозволяє центральному процесору керувати потужними виконавчими механізмами за допомогою стандартних логічних сигналів GPIO. З метою забезпечення масштабованості архітектури передбачено вузол розширення портів, який надає додаткові лінії вводу-виводу для підключення нової периферії. Крім того, функціонал модуля може бути опціонально розширений засобами візуальної індикації (графічним дисплеєм) для відображення поточного стану системи та мережевих параметрів.

Підсистема живлення забезпечує автономне функціонування пристрою, включає блок контролю заряду акумуляторної батареї, схему захисту та стабілізатори напруги, що формують необхідні рівні живлення для цифрової та аналогової частин схеми.

Інформаційні потоки організовані таким чином мікроконтролерний блок є ведучим пристроєм на шинах даних, ініціюючи запити до сенсорів та пам'яті. Отримані дані збираються, обробляються та передаються через бездротовий інтерфейс до хмарного середовища або, у разі аварії мережі, зберігаються локально. Така архітектура забезпечує гнучкість, надійність та можливість модернізації окремих вузлів без зміни загальної структури пристрою.

3.2 Вибір та обґрунтування елементної бази

Вибір та обґрунтування елементної бази є визначальним етапом проектування апаратної частини, оскільки він безпосередньо впливає на функціональність, енергоефективність, габарити та кінцеву вартість

розроблюваного модуля. Цей вибір проводився на основі теоретичного аналізу та поставлених завдань. Процес вибору декомпозовано на три основні групи: вибір центрального мікроконтролера, вибір підсистеми збору даних та вибір периферійних інтерфейсів керування.

3.2.1 Мікроконтролерний блок

Проектування апаратного забезпечення будь-якого сучасного IoT-пристрою починається з вибору центрального мікроконтролера (МК), який є його «серцем» і визначає функціональні можливості, продуктивність та потенціал для подальшого масштабування усієї системи. На основі аналізу, проведеного в Розділі 1, було сформульовано низку ключових вимог до базової платформи [28].

З урахуванням цих вимог, у якості базової платформи для розробки універсального мікропроцесорного модуля було обрано Arduino Nano ESP32. Цей вибір обґрунтовується тим, що дана плата базується на одній з найсучасніших систем-на-кристалі (SoC) від Espressif Systems — ESP32-S3, яка поєднує високу продуктивність, багатий набір периферії та розширені можливості [8].

Ключовою архітектурною перевагою є висока продуктивність та підтримка багатозадачності. Пристрій повинен одночасно зчитувати дані з кількох сенсорів, обробляти логіку керування виконавчими механізмами (MOSFET-виходи) та підтримувати активне мережеве з'єднання (Wi-Fi/MQTT).

На відміну від одноядерних архітектур (як ESP8266), де мережевий стек неминуче конфліктує з прикладним кодом, двоядерний процесор Xtensa 32-bit LX7 в ESP32-S3 дозволяє повністю уникнути цієї проблеми. Використання операційної системи реального часу FreeRTOS (теоретичні основи якої розглянуто в 2.4) дає змогу «прив'язати» ресурсоємні завдання мережевої комунікації до одного ядра (PRO_CPU), тоді як друге ядро (APP_CPU)

залишається повністю вільним для виконання критичного до часу коду користувача [9].

Другою важливою вимогою є підтримка гібридного бездротового зв'язку. Інтегрований в ESP32-S3 радіомодуль, що підтримує як Wi-Fi 802.11 b/g/n, так і Bluetooth Low Energy (BLE) 5.0, надає таку гнучкість. Wi-Fi використовується як основний канал для високошвидкісної передачі даних до хмарного брокера MQTT. Водночас BLE слугує для вирішення двох важливих завдань: початкового налаштування пристрою (provisioning) та локального керування/моніторингу на короткій відстані. Наявність обох технологій на одному кристалі значно спрощує схемотехніку.

Третьою, і, можливо, найважливішою вимогою для автономних пристроїв є енергоефективність. Платформа ESP32-S3 пропонує гнучку систему управління живленням з кількома режимами сну. Окрім стандартного режиму Deep Sleep, ESP32-S3 має ULP (Ultra-Low-Power) копроцесор. Цей допоміжний процесор може працювати, поки основні ядра перебувають у сні, виконуючи прості завдання, наприклад, моніторинг показників АЦП або перевірку стану GPIO [8].

Нарешті, ESP32-S3 пропонує багатий набір периферії та апаратну безпеку. Він має до 45 програмованих виводів GPIO (рис. 3.1), високоточні 12-бітні АЦП, а також апаратні криптографічні прискорювачі для алгоритмів AES, SHA та RSA, що є критичним для реалізації захищених TLS-з'єднань [9]. Форм-фактор плати Arduino Nano ESP32 є компактним та зручним для макетування. Таким чином, сукупність цих характеристик робить платформу Arduino Nano ESP32 оптимальним та обґрунтованим вибором.

3.2.2 Підсистема збору даних

Після вибору мікроконтролерного блоку, наступним кроком є вибір периферії для збору та зберігання даних. Для забезпечення гнучкості та мінімізації кількості задіяних виводів (GPIO) мікроконтролера, перевага

надавалася пристроям зі стандартизованими цифровими шинами I2C та SPI (теоретичні основи яких розглянуто в 2.3).

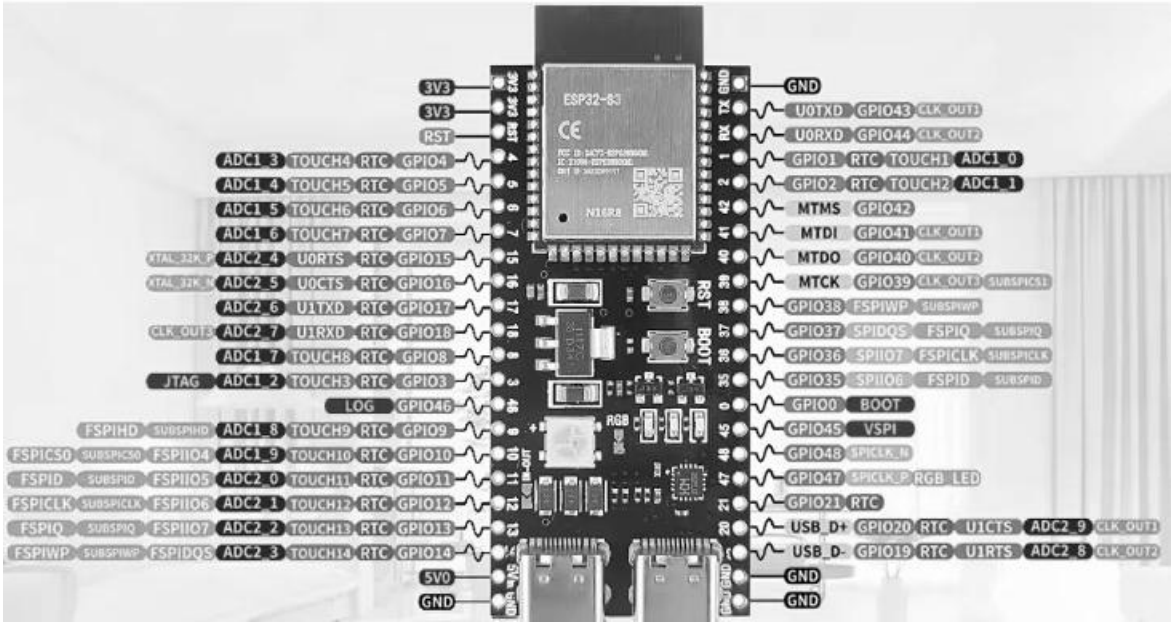


Рисунок 3.1 — Схема інформаційних потоків та взаємодії компонентів модуля

Для комплексного моніторингу кліматичних параметрів було обрано цифровий сенсор BME280 від Bosch Sensortec (рис. 3.2). Ця ІМС є високоінтегрованим рішенням, що об'єднує на одному кристалі датчики температури, барометричного тиску та відносної вологості. BME280 забезпечує високу точність завдяки заводському калібруванню та підтримує інтерфейс I2C [29]. Важливою особливістю BME280 є його гнучке управління енергоспоживанням, зокрема режим Forced mode, який дозволяє мікроконтролеру "пробудити" сенсор, виконати одне вимірювання та негайно повернути його в стан сну, що ідеально узгоджується з завданням розробки адаптивного алгоритму енергозбереження [29].

Для реалізації функцій адаптивного керування освітленням необхідний точний датчик рівня освітленості. Використання простих аналогових фоторезисторів (LDR) є неприйнятним через їх значну нелінійність та температурну залежність. Тому було обрано цифровий сенсор BH1750 від Rohm

Semiconductor. Це калібрований цифровий пристрій з інтерфейсом I2C, ключовою перевагою якого є спектральна чутливість, близька до кривої чутливості людського ока [30]. Це дозволяє модулю вимірювати освітленість у люксах та об'єктивно реагувати на рівень освітлення. Зовнішній вигляд датчика освітленості BH1750 наведено на рисунку 3.3.

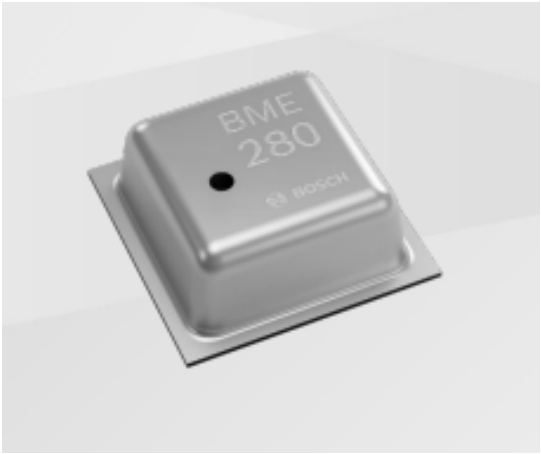


Рисунок 3.2 — Зовнішній вигляд інтегрального сенсора BME280 [29]

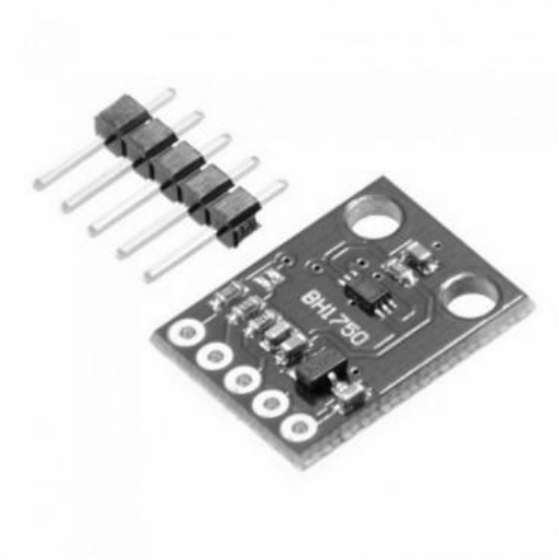


Рисунок 3.3 — Зовнішній вигляд датчика освітленості BH1750

3.2.3 Підсистема хронометрії та зберігання даних

Для будь-якого IoT-пристрою, що виконує функції логування, критично важливою є наявність точних часових міток (timestamps). Вбудований RTC

мікроконтролера ESP32-S3 є неточним та втрачає хід часу при повному знеструмленні. Для забезпечення повної автономності та надійної фіксації часу було обрано зовнішній модуль годинника реального часу DS3231 [31].

Цей пристрій є надзвичайно точним I²C годинником, оскільки містить вбудований термокомпенсований кварцовий резонатор (TCXO). Він автоматично коригує частоту генератора залежно від температури, забезпечуючи виняткову точність (±2 хвилини на рік). Наявність входу для резервної батареї (CR2032) гарантує безперервну роботу годинника протягом років, навіть якщо основне живлення модуля вимкнене [31]. Зовнішній вигляд годинника реального часу показано на рисунку 3.4.

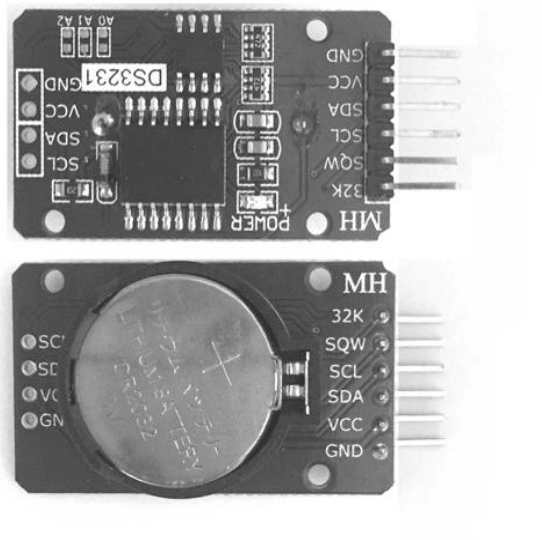


Рисунок 3.4 — Зовнішній вигляд модуля годинника реального часу DS3231

Для запобігання втраті даних у разі відсутності Wi-Fi з'єднання, в архітектурі модуля передбачено локальне сховище даних на базі карти пам'яті MicroSD. Вони підключаються до мікроконтролера Nano ESP32 через високошвидкісну шину SPI. Використання стандартної файлової системи, такої як FAT32, дозволяє не лише накопичувати великі архіви даних, але й легко зчитувати їх на будь-якому ПК [32].

3.2.4 Підсистема керування

Окрім моніторингу, універсальний модуль повинен керувати зовнішніми пристроями. Для керування силовими навантаженнями (вентилятори, освітлення) будемо використовувати MOSFET-ключі. Цифрові виводи (GPIO) мікроконтролера Nano ESP32 є слабкострумовими (до 40 мА) і працюють з логікою 3.3 В, чого недостатньо для керування реальними виконавчими механізмами [33]. Для узгодження рівнів потужності використовується N-канальний MOSFET з логічним рівнем керування (Logic-Level Gate), такий як IRLZ44N. Такі транзистори гарантовано повністю відкриваються вже при подачі 3.3 В на затвор, на відміну від стандартних, яким потрібно 10-12 В. MOSFET вмикається у "низький ключ" (low-side switch), керуючи підключенням навантаження до землі (GND) [33].

Хоча ESP32-S3 має 45 GPIO, на компактній платі Nano ESP32 більшість з них задіяні. Для гнучкого розширення можливостей вводу-виводу (підключення кнопок, індикаторів) використовується розширювач портів MCP23017. Ця мікросхема підключається до шини I2C і надає додаткові 16 ліній GPIO. Важливою перевагою є підтримка апаратних переривань (Interrupts), що дозволяє реалізувати асинхронну обробку подій (наприклад, натискання кнопки), не змушуючи ESP32 постійно опитувати (poll) стан пінів.

3.3 Розробка функціональної схеми універсального модуля

Розробка функціональної схеми є фінальним етапом апаратного проектування, що переводить функціональну архітектуру у площину конкретної інженерної реалізації. Схема є фундаментальним документом, що визначає логіку роботи пристрою. Вона графічно представляє всі електронні компоненти та всі електричні з'єднання між їхніми виводами у вигляді іменованих мереж (nets) [34].

Розробку функціональної схеми універсального мікропроцесорного модуля розпочнемо з розробки схеми підсистеми живлення, яка повинна

вирішувати такі три основні задачі:

— забезпечення тривалої роботи від акумулятора;

— можливість безпечної зарядки акумулятора від стандартного джерела (USB);

— формування стабільної напруги живлення (3.3 В) для всіх компонентів модуля.

Як джерело автономного живлення будемо використовувати літій-полімерний (Li-Po) акумулятор. Цей тип акумуляторів має найвищу питому енергетичну густину серед доступних на ринку, низький саморозряд та широкий діапазон робочих температур [35]. Номінальна напруга одного Li-Po елемента становить 3.7 В, однак його реальна робоча напруга змінюється у широкому діапазоні: від 4.2 В (повний заряд) до ~3.0 В (повний розряд). Такий нестабільний діапазон напруги створює дві схемотехнічні проблеми.

Перша проблема — необхідність безпечної зарядки. Для Li-Po акумуляторів потрібен суворий профіль заряду CC/CV (Constant Current / Constant Voltage): спочатку зарядка постійним струмом до досягнення 4.2 В, а потім підтримка постійної напруги 4.2 В до падіння струму заряду до мінімального. Для реалізації цього профілю та забезпечення захисту від перезаряду, обов'язковим є використання спеціалізованої інтегральної схеми (IC) — контролера заряду. Для даного проекту обрано IC серії TP4056 (або аналогічну BQ2057TTS), що живиться безпосередньо від +5 В порту USB, забезпечує коректний профіль CC/CV та має виходи для індикації стану заряду.

Мікросхема TP4056 є повноцінним лінійним зарядним пристроєм постійного струму/постійної напруги для літій-іонних акумуляторів. Завдяки внутрішній архітектурі PMOSFET блокувальний діод не потрібен, а схема запобігає негативному струму заряду. Термічний зворотний зв'язок регулює струм заряду, обмежуючи температуру кристала під час роботи з високою потужністю або високої температури навколишнього середовища. Напруга заряду фіксована на рівні 4,2 В, а струм заряду можна програмувати зовні за допомогою одного резистора. TP4056 автоматично завершує цикл заряду, коли

струм заряду падає до 1/10 запрограмованого значення після досягнення кінцевої напруги підтримуючого заряду. Інші функції TP4056 включають монітор струму, блокування при зниженій напрузі, автоматичне перезаряджання та два контакти стану для індикації завершення заряду та наявності вхідної напруги.

Типова схема вмикання мікросхеми TP4056 наведена на рисунку 3.5.

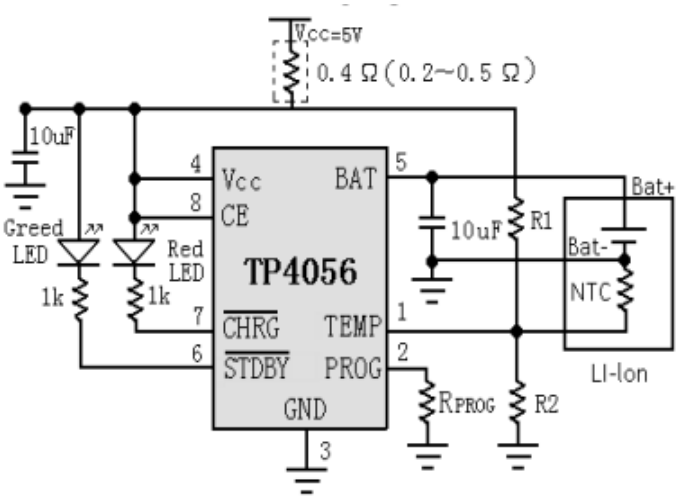


Рисунок 3.5 — Типова схема вмикання інтегрального лінійного зарядного пристрою TP4056

Друга, і більш складна, проблема — це необхідність перетворення нестабільної напруги акумулятора у стабільні 3.3 В, які необхідні для живлення мікроконтролера ESP32-S3 та абсолютної більшості цифрових сенсорів. Для цього завдання можна використовувати два типи перетворювачів: імпульсні (SMPS) або лінійні (LDO) [36].

Імпульсні перетворювачі (SMPS), такі як понижуючі (Buck) або підвищуючо-понижуючі (Buck-Boost), є дуже ефективними. Вони мають високий ККД (коефіцієнт корисної дії), який часто перевищує від 85 до 95%, незалежно від різниці вхідної та вихідної напруги. Однак, незважаючи на цю перевагу, вони мають фундаментальний недолік для нашого проекту: за принципом своєї роботи вони генерують високочастотні електромагнітні завади

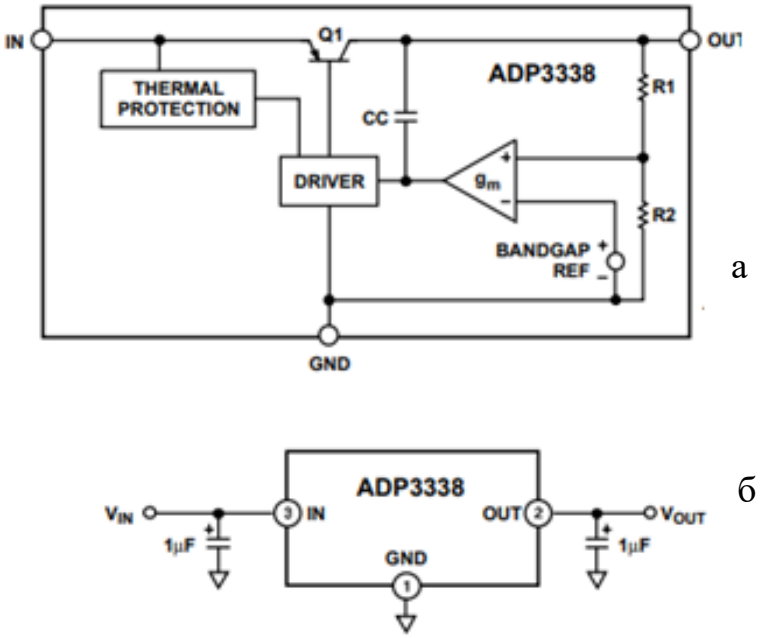
(EMI). Ці завади (шум) у лінії живлення можуть негативно впливати на чутливі аналогові компоненти (якщо вони є) та, що особливо важливо, на радіочастотну частину (RF) самого модуля ESP32. Радіошум може значно погіршити чутливість приймача та стабільність Wi-Fi та Bluetooth зв'язку [37].

Тому для даного проекту було обрано лінійний стабілізатор (LDO - Low-Dropout). LDO є значно простішими, дешевшими та не створюють жодних високочастотних завад, забезпечуючи дуже "чисте" живлення. Їхній головний недолік — нижчий ККД, оскільки вони розсіюють "зайву" різницю напруги (V_{in} - V_{out}) у вигляді тепла. Ефективність LDO розраховується за прямою формулою: ККД = V_{out} / V_{in}. У нашому випадку, при повністю зарядженому акумуляторі (4.2 В), ККД становитиме 3.3 В / 4.2 В ≈ 78.5%. Коли акумулятор розряджається (наприклад, до 3.7 В), ККД зростає до 3.3 В / 3.7 В ≈ 89.2%.

На перший погляд, це менш ефективно, ніж 90% у SMPS. Однак, в нашому IoT-пристрої модуль 99% часу перебуватиме в режимі Deep Sleep (глибокого сну), де загальне споживання складає мікроампери. У цьому режимі ККД перетворення майже не має значення. Натомість на перше місце виходить інший параметр — власний струм спокою (Quiescent Current, Iq) стабілізатора. Це той струм, який LDO споживає "сам на себе", просто щоб бути увімкненим. У LDO типу ADP3338 (або аналогічних) цей струм надзвичайно низький. У той час як SMPS часто мають значно вищий струм спокою, який може стати основним джерелом розряду батареї під час сну. Таким чином, для пристроїв, що працюють у режимі сну, LDO з низьким Iq є оптимальним вибором.

Лінійний стабілізатор ADP3338, функціональна схема та схема вмикання якого наведені на рисунку 3.6, забезпечує струм навантаження до 1 А. Серед традиційних LDO ADP3338 виділяється новою архітектурою та вдосконаленим процесом, який пропонує переваги в продуктивності та вищий вихідний струм, ніж у конкурентів. Його запатентована конструкція вимагає лише вихідного конденсатора ємністю 1 мкФ для стабільності. Цей пристрій нечутливий до еквівалентного послідовного опору вихідного конденсатора та стабільно працює з будь-якими якісними конденсаторами, включаючи керамічні, для застосувань

з обмеженим простором. ADP3338 досягає виняткової точності ±0,8% за кімнатної температури та ±1,4% при змінах температури, мережі та навантаження. Напруга падіння ADP3338 становить лише 190 мВ при 1 А. Пристрій також має безпечне обмеження струму та захист від теплового перевантаження. ADP3338 має наднизький струм спокою: 110 мкА в умовах невеликого навантаження.



а

б

Рисунок 3.6 — Функціональна схема (а) та схема вмикання (б) лінійного стабілізатора ADP3338

Ключовим параметром LDO є напруга падіння (Dropout Voltage) — мінімальна різниця між V_{in} та V_{out}, при якій стабілізатор ще може підтримувати 3.3 В на виході. Оскільки мікроконтролер ESP32-S3 здатний стабільно працювати при падінні напруги живлення аж до 3.0 В [8], обраний LDO з наднизькою напругою падіння забезпечить стабільну роботу системи, повністю використовуючи весь діапазон напруги акумулятора.

Для забезпечення стабільної роботи високочастотних цифрових мікросхем, схема живлення потребує ретельної фільтрації. Кожна мікросхема на платі (ESP32, BME280, DS3231 та ін.) отримує власний локальний фільтруючий

(розв'язуючий) конденсатор (зазвичай керамічний 100 нФ), який розміщується фізично якомога ближче до її виводів живлення VCC/GND. Додатково, на основній шині +3.3 В (на виході LDO) встановлюється танталовий або керамічний конденсатор великої ємності (від 10 до 22 мкФ) для згладжування низькочастотних пульсацій та забезпечення пікового споживання струму (наприклад, під час активації Wi-Fi Tx) [38].

Управління електричним у режимі вмикання/вимикання зазвичай здійснюється за допомогою електромагнітних реле. Управління навантаженням за допомогою реле є це одним з найпростіших і найнадійніших способів комутації потужних високовольтних кіл постійного та змінного струму за допомогою низькострумоових сигналів. Для керування реле використовується транзисторний ключ на польовому транзисторі (рисунок 3.7).

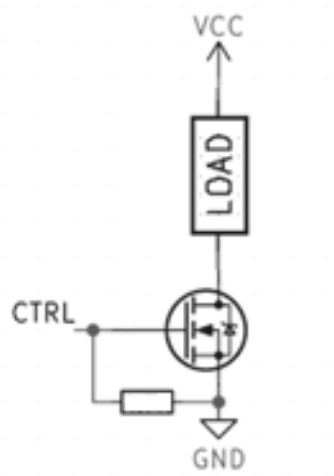


Рисунок 3.7 — Управління навантаженням за допомогою транзисторного ключа

З врахуванням викладеного вище була розроблена функціональна схема універсального мікропроцесорного модуля, яка наведена у Додатку Г. Головним елементом мікропроцесорного модуля є модуль ESP32-S3 DD5, в основі якого лежить потужний двоядерний мікроконтролер Xtensa LX7 з підтримкою бездротових 2.4 ГГц технологій зв’язку Wi-Fi (802.11 b/g/n) та Bluetooth 5 (LE).

До ліній GPIO4 та GPIO5 мікроконтролера DD5, альтернативною функцією яких є лінії SCL та SDA шини I2C, підключені цифровий скесор температури, вологості та атмосферного тиску BME280 DD1, сенсор освітленості BH1750 DD6 та модуль годинника реального часу DS3231 DD2. Решта вузлів функціональної схеми реалізують з'єднання, обрані в 3.2. Для коректної роботи протоколу I2C (який працює за логікою "відкритого колектора") на схемі обов'язково встановлюються два зовнішні підтягуючі резистори (pull-up resistors) номіналом 4.7 кОм від ліній SDA та SCL до шини живлення +3.3 В.

Лінії введення/виведення GPIO13, GPIO11 та GPIO12 мікроконтролера DD5 утворюють відповідно лінії MISO, MOSI та SCK шини SPI, що використовується для з'єднання мікроконтролера з модулем MicroSD-карти DD7. Для адресації MicroSD-карти використовується індивідуальна лінія вибору кристала (Chip Select, CS), що підключена до виводу GPIO10.

Вузол керування навантаженнями реалізований на N-канальних MOSFET VT1 та VT2 з логічним рівнем керування, включених за схемою "низький ключ" (low-side switch). Схема керування затвором (Gate) транзисторів, що підключені до GPIO3 та GPIO9, містить pulldown резистори R2 та R3 номіналом 10 кОм, які гарантують підтримання транзисторів у закритому стані під час перезавантаження мікроконтролера. Підключення навантаження до універсального модуля здійснюється через роз’єми X3 та X4. У схемі не передбачена наявність реле. Вибір реле залишається за кінцевим користувачем, який буде вибирати реле відповідно до потужності навантаження, яким планується керувати.

На елементах DD3 та DD4 зібрана підсистема живлення, яка забезпечує живлення електричної схеми універсального мікропроцесорного модуля від зовнішнього джерела постійної напруги +5В, що підключається через роз’єм X1, або від акумуляторної батареї, що підключається через роз’єм X2. Крім того, підсистема живлення забезпечує зарядження акумулятора, коли пристрій підключений до зовнішнього джерела +5В. Інтегральний лінійний зарядний

пристрій TP4056 DD3 керує процесом зарядження акумулятора. Лінійний інтегральний стабілізатор ADP3338 DD4 забезпечує формування стабільної напруги +3,3 В, яка необхідна для живлення цифрових елементів схеми.

Для розширення можливостей застосування універсального модуля в схемі закладена можливість використання шістьох ліній введення/виведення від GPIO33 до GPIO38 та асинхронного послідовного каналу зв'язку, що включає лінії TxD та RxD. Фізичне підключення зовнішніх компонентів до цих лінії здійснюється через роз'єм X3.

3.4 Розробка програмного забезпечення модуля

Розробка програмного забезпечення (firmware) універсального вимірювального модуля здійснювалася мовою C++ з використанням фреймворку Arduino. Такий вибір зумовлений широкою підтримкою апаратних бібліотек, доступністю драйверів для вибраних сенсорів та повною сумісністю з архітектурою мікроконтролера ESP32-S3 [8]. Крім того, Arduino-фреймворк містить вбудовану інтеграцію з механізмами FreeRTOS [24], що дає змогу реалізувати багатозадачність та ефективний асинхронний обмін даними.
Архітектура прошивки побудована за модульним принципом і включає:

— ініціалізацію периферійних модулів;
— конфігурацію мережевих інтерфейсів;
— обробку команд керування;
— формування та передачу телеметричних даних;
— використання черг повідомлень FreeRTOS для ізоляції задач.

Для розробки програмного забезпечення використовувалося Arduino IDE [9]. Для роботи з контролером ESP32-S3 у середовище встановлено пакет «esp32 by Espressif Systems», що містить компілятор Xtensa, SDK-компоненти та інструменти, описані в офіційній документації ESP-IDF [24].

Перед початком програмування у меню Tools → Board Manager обрано цільову платформу «Arduino Nano ESP32» [9]. Також активовано параметр USB-

CDC, що забезпечує коректний вивід діагностичних повідомлень через послідовний інтерфейс (рисунок 3.8).

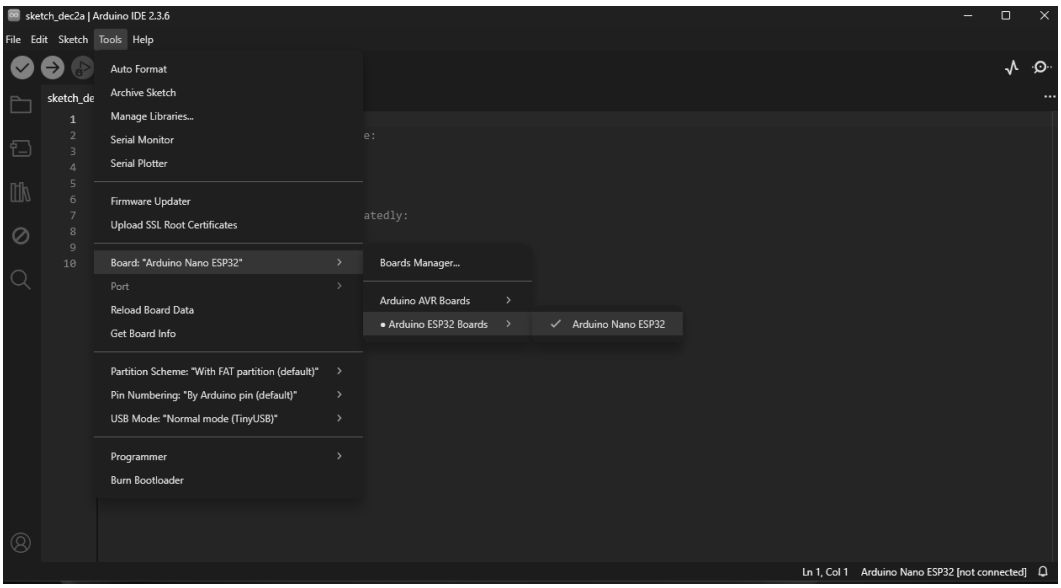


Рисунок 3.8 — Вибір цільової плати Arduino Nano ESP32 у середовищі Arduino IDE

Для взаємодії з апаратними компонентами, визначеними у підрозділі 3.2, використовуються спеціалізовані бібліотеки, що забезпечують доступ до сенсорів та мережевої інфраструктури на високому рівні абстракції.

Основні заголовні файли наведено у лістингу 3.1.

Лістинг 3.1 — Підключення бібліотек та оголошення об’єктів

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <Adafruit_BME280.h>
#include <BH1750.h>
#include <ArduinoJson.h>
Adafruit_BME280 bme;
BH1750 lightMeter;
WiFiClientSecure espClient;
PubSubClient client(espClient);
```

Бібліотека WiFiClientSecure забезпечує встановлення TLS/SSL-з’єднання, що є критично важливим для захисту даних у сучасних IoT-системах [7].

Для підключення до Wi-Fi та взаємодії з MQTT-брокером оголошено набір глобальних констант. Структура MQTT-топіків формувалася ієрархічно, що спрощує маршрутизацію потоків телеметрії та команд керування [19]. Конфігурація показана у лістингу 3.2.

Лістинг 3.2 — Налаштування реквізитів мережі та MQTT-топіків

```
const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 8883;
const char* topic_data = "device/nano_esp32/sensors";
const char* topic_control = "device/nano_esp32/control";
```

Застосування порту 8883 дає змогу використовувати шифрування MQTT-трафіку, що відповідає сучасним вимогам до безпеки IoT-комунікацій [7].

Протокол MQTT використовує асинхронну модель обміну, тому обробка вхідних повідомлень виконується у функції callback. Вона автоматично запускається при отриманні даних у топіку device/nano_esp32/control. Алгоритм включає десеріалізацію повідомлення та керування цифровими виходами (GPIO) [8].

Лістинг 3.3 — Callback-функція для обробки команд

```
void callback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
  if (String(topic) == topic_control) {
    if (message == "RELAY_ON") {
      digitalWrite(RELAY_PIN, HIGH);
    }
    else if (message == "RELAY_OFF") {
      digitalWrite(RELAY_PIN, LOW);
    } } }
```

Основний робочий цикл пристрою побудований на базі FreeRTOS, що забезпечує виконання задач у паралельних потоках [24]. Зчитування сенсорів

BME280 [29] та BH1750 [30] виконується у таймерній задачі, що виключає блокування мережевого стеку. Отримані дані упаковуються у JSON-формат за допомогою бібліотеки ArduinoJson, що спрощує подальшу інтеграцію з хмарними сервісами та серверними застосунками [5]. Блок-схему принципу роботи програмного забезпечення універсального мікропроцесорного модуля наведено в Додатку Д.

Лістинг 3.4 — Формування та надсилання JSON-телеметрії

```
void sendTelemetry() {
  JsonDocument doc;
  doc["temperature"] = bme.readTemperature();
  doc["humidity"] = bme.readHumidity();
  doc["pressure"] = bme.readPressure() / 100.0F;
  doc["lux"] = lightMeter.readLightLevel();
  char buffer[256];
  serializeJson(doc, buffer);
  client.publish(topic_data, buffer);
}
```

Такий підхід забезпечує структуроване, уніфіковане та легке для обробки представлення телеметричних даних у системах моніторингу та керування. Повний лістинг програмного коду універсального мікропроцесорного модуля наведено в Додатку Е.

4 ВВЕДЕННЯ В ЕКСПЛУАТАЦІЮ ТА НАЛАШТУВАННЯ МІКРОПРОЦЕСОРНОГО МОДУЛЯ

4.1 Апаратна конфігурація та порядок підключення модуля

Етап апаратної конфігурації є критичним для забезпечення надійності подальшої експлуатації пристрою. Він включає перевірку компонентної бази, фізичний монтаж елементів на друковану плату (або макетну плату типу breadboard) та контроль електричних параметрів перед першим запуском.

Перед початком монтажу необхідно провести вхідний контроль компонентів. Особливу увагу слід приділити центральному модулю Arduino Nano ESP32. Згідно з технічною документацією [9], робоча напруга логічних рівнів мікроконтролера становить 3.3 В. Подача напруги 5 В на вхідні піни (GPIO) може призвести до незворотного пошкодження кристала ESP32-S3 [8]. Тому необхідно переконатися, що всі зовнішні модулі сумісні з логікою 3.3 В.

Монтаж починається з ланцюгів живлення, оскільки стабільність напруги є запорукою коректної роботи цифрової частини.

До роз'єму X2 підключається літій-полімерний (Li-Po) акумулятор. Необхідно суворо дотримуватися полярності: червоний провід — до клеми «+» (BAT+), чорний — до клеми «-» (GND). За допомогою мультиметра вимірюється напруга на виході стабілізатора LDO. Вона має знаходитися в діапазоні від 3.25 В до3.35 В незалежно від рівня заряду акумулятора [37].

Під час виконання монтажних операцій не допускається робота з підключеним акумулятором, щоб уникнути коротких замикань.

Для підключення зовнішніх виконавчих механізмів та додаткового обладнання передбачено роз'єми X3, X4 та X5. Підключення силового навантаження здійснюється до роз'ємів X3 та X4. Ці виходи комутуються через вбудовані MOSFET-ключі, що дозволяє керувати потужними споживачами безпосередньо, без використання додаткових драйверів. Під час роботи з індуктивними навантаженнями (електродвигуни, соленоїди) рекомендується

перевіряти правильність полярності підключення та забезпечувати надійний контакт, що сприяє стабільній роботі комутаційного каналу.

Для розширення функціональних можливостей модуля використовується багатоцільовий роз'єм X5. Він забезпечує доступ до вільних ліній введення-виведення (GPIO) мікроконтролера, а також до шин живлення (+3.3 В, GND). Лінії цього роз’єму можуть бути програмно сконфігуровані відповідно до вимог конкретного проєкту, що дозволяє підключати додаткові сенсори, комунікаційні модулі або інтегрувати пристрій у складніші системи автоматизації без серйозних змін. Наявність декількох інтерфейсних вузлів підвищує гнучкість апаратної конфігурації та дає можливість масштабувати систему у разі розширення її функціоналу.

Перед подачею живлення проводиться візуальний огляд плати на предмет відсутності перемичок припою (short circuits) та перевірка правильності орієнтації мікросхем (ключів). Після увімкнення живлення тумблером SW1 необхідно проконтролювати статусний світлодіод на платі Arduino Nano ESP32:

— світіння зеленого індикатора «PWR» свідчить про наявність коректного живлення 3.3 В;

— короткочасне миготіння вбудованого помаранчевого світлодіода (Pin 13) свідчить про успішний старт завантажувача (Bootloader).

4.2 Налаштування програмного забезпечення та підключення до хмарного середовища IoT

Після верифікації апаратної частини ключовим етапом введення в експлуатацію є налаштування програмної взаємодії в межах гетерогенної мережі. Враховуючи вимоги до універсальності модуля, було реалізовано гібридну архітектуру обміну даними, яка поєднує переваги протоколів MQTT та HTTP. Такий підхід узгоджується з принципами побудови розподілених систем, де інтеграція різнорідних протоколів дозволяє компенсувати недоліки кожного з них та забезпечити комплексну взаємодію між рівнями Edge та Cloud в умовах

обмежених ресурсів [39].

Принцип функціонування системи Логіка роботи програмного забезпечення базується на паралельному виконанні двох мережевих завдань. Канал оперативного керування (Real-time Control) реалізований на базі протоколу MQTT. Цей протокол забезпечує мінімальну затримку (latency) та дозволяє миттєво передавати команди від смартфона користувача до виконавчих механізмів модуля (реле/MOSFET) [19]. Канал аналітики (Data Logging) реалізований через RESTful API та протокол HTTP. Він використовується для періодичної (раз на 20 с) відправки пакетів телеметрії на сервер аналітики для побудови графіків та довгострокового зберігання [36].

Налаштування мобільного застосунку Для забезпечення зручного інтерфейсу користувача (Human-Machine Interface) використовується мобільний застосунок «IoT MQTT Panel», доступний для завантаження у магазинах цифрової дистрибуції (рис. 4.1). Цей інструментарій дозволяє створювати кастомізовані панелі керування (Dashboards) без необхідності написання власного коду під Android/iOS.

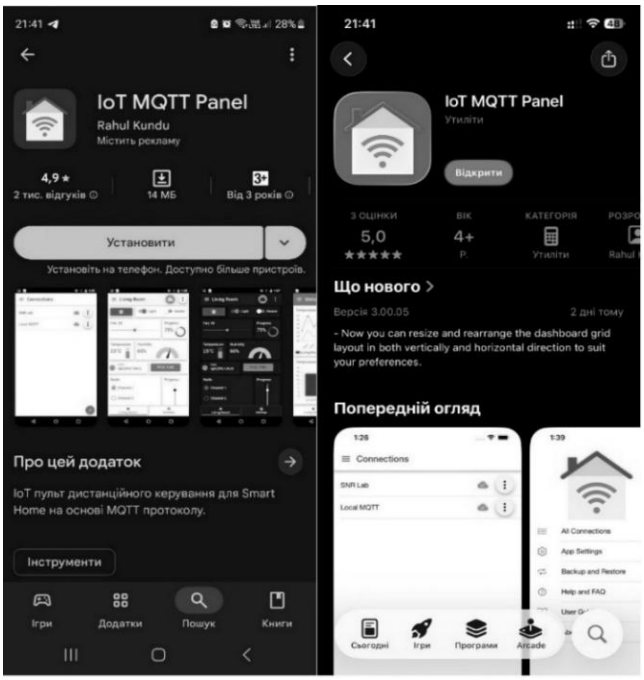


Рисунок 4.1 — Сторінка завантаження клієнтського застосунку «IoT MQTT Panel» у магазині додатків

Процес налаштування мобільного застосунку для взаємодії з розробленим IoT-модулем починається зі створення нового профілю підключення. Після запуску програми користувачу відображається повідомлення про відсутність активних MQTT-з’єднань, якщо це не новий користувач, присутня кнопка додавання нових з’єднань. (рис. 4.2).

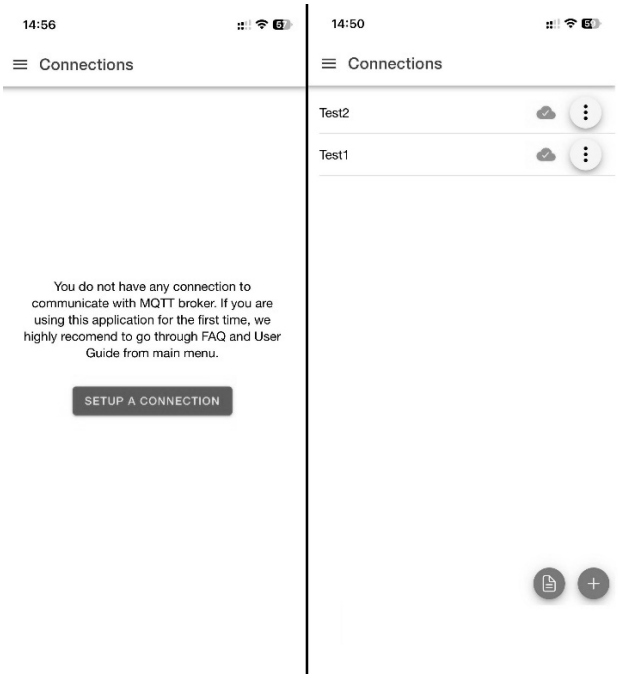


Рисунок 4.2 — Початковий екран застосунку перед та після налаштуваннь MQTT-з’єднання

Для продовження необхідно натиснути кнопку “Setup a connection”, після чого відкривається форма додавання нового підключення (рис. 4.3). У цій формі задається назва підключення, після чого вводяться параметри брокера.

У полі Broker address вказується домен MQTT-сервера — у тестовій конфігурації використано публічний брокер broker.hivemq.com. Стандартний порт для незашифрованого з’єднання — 1883, тому він встановлюється за замовчуванням, а як транспортний протокол обирається TCP. Після підтвердження налаштувань профіль з’єднання з’являється у списку доступних підключень.

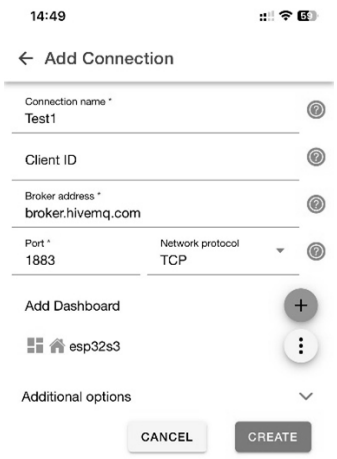


Рисунок 4.3 — Форма створення нового MQTT-підключення

Після створення з’єднання користувач переходить до панелі візуалізації (Dashboard). Якщо панель ще порожня, застосунок пропонує створити перший елемент інтерфейсу за допомогою кнопки «Add a panel». Список доступних елементів містить різні типи віджетів, серед яких елементи керування, текстові поля, індикатори та шкали (рис. 4.4).

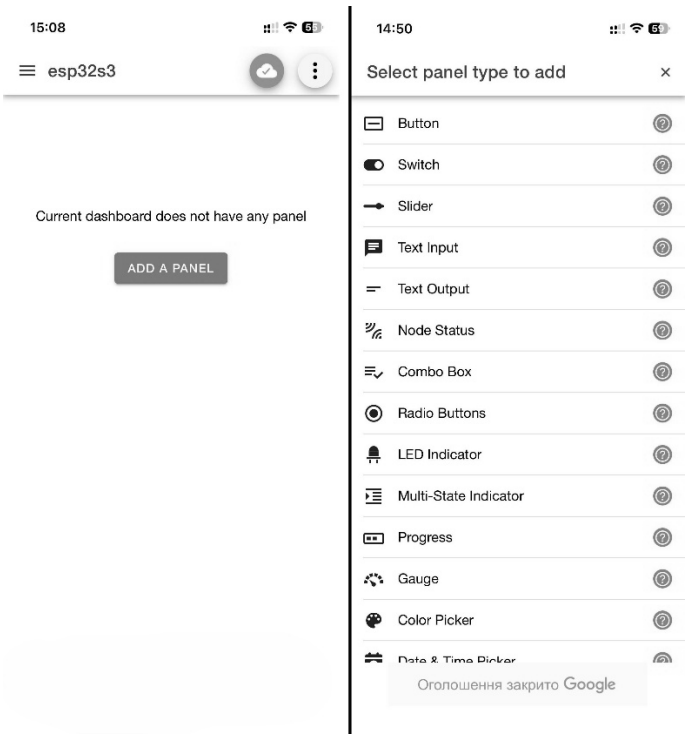


Рисунок 4.4 — Порожня панель Dashboard, та список віджетів

Для відображення температури було обрано віджет типу Gauge, який дозволяє зручно візуалізувати значення у вигляді кругової шкали. У вікні налаштування віджета (рис. 4.5) задається його назва, після чого в полі топіка вказується шлях до MQTT-каналу, у якому модуль публікує дані сенсорів. У даній роботі використовується топік «device/nano_esp32_test/sensors»ю.

Оскільки пристрій передає значення у форматі JSON, застосунок підтримує вибір конкретного поля всередині пакета. Для цього у параметрі JsonPath for subscribe зазначається шлях до потрібної величини, наприклад $.temp для температури. Таким чином віджет автоматично виділяє необхідний параметр з отриманого повідомлення.

Крім того, задається діапазон значень, у якому працює індикатор (наприклад, від 0 до 35 °C), а також пороги зміни кольору шкали, що дозволяє візуально відстежувати нормальні та критичні режими роботи сенсорів. Після збереження налаштувань віджет починає оновлюватися у режимі реального часу при кожному надходженні MQTT-повідомлення від модуля.

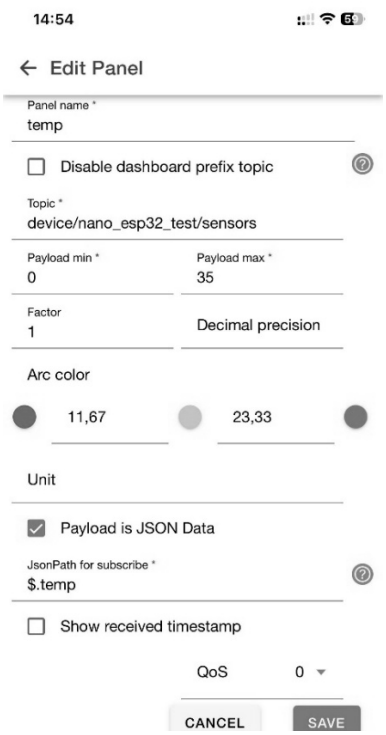


Рисунок 4.5 — Налаштування віджета типу “Gauge” для відображення температури.

Результат налаштування інтерфейсу на мобільному пристрої, що демонструє отримання даних у реальному часі, наведено на рисунку 4.6.

Інтеграція з веб-сервісом аналітики ThingSpeak використовується для вирішення задачі накопичення даних та їх візуалізації у веб-браузері було використано платформу ThingSpeak. На відміну від MQTT, цей сервіс дозволяє зберігати історію вимірювань, що є критичним для аналізу кліматичних трендів [19].

Для організації хмарної аналітики та зберігання телеметричних даних було використано сервіс ThingSpeak, який надає можливість створення окремих каналів для кожного пристрою та побудови графіків параметрів у реальному часі.

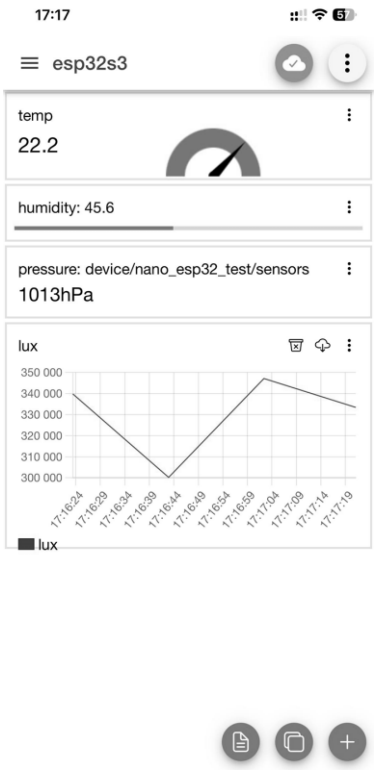


Рисунок 4.6 — Інтерфейс мобільного додатку для оперативного моніторингу та керування

Робота із сервісом починається зі створення облікового запису, після чого користувач отримує доступ до панелі керування каналами. На етапі конфігурації

було створено новий канал, у якому визначено чотири поля, що відповідають параметрам, які передає модуль: Temperature, Humidity, Pressure та Light. Інтерфейс базових налаштувань каналу наведено на рисунку 4.7. У цьому вікні задається його назва, а також визначається кількість та призначення інформаційних полів, які згодом використовуються для побудови графічних візуалізацій.

Після збереження конфігурації канал отримує унікальний ідентифікатор Channel ID, що використовується мікроконтролером для адресації HTTP-запитів. Для авторизації запису даних сервіс автоматично генерує спеціальні API-ключі. На стороні пристрою використовується Write API Key, який забезпечує можливість надсилати дані у закритий приватний канал.

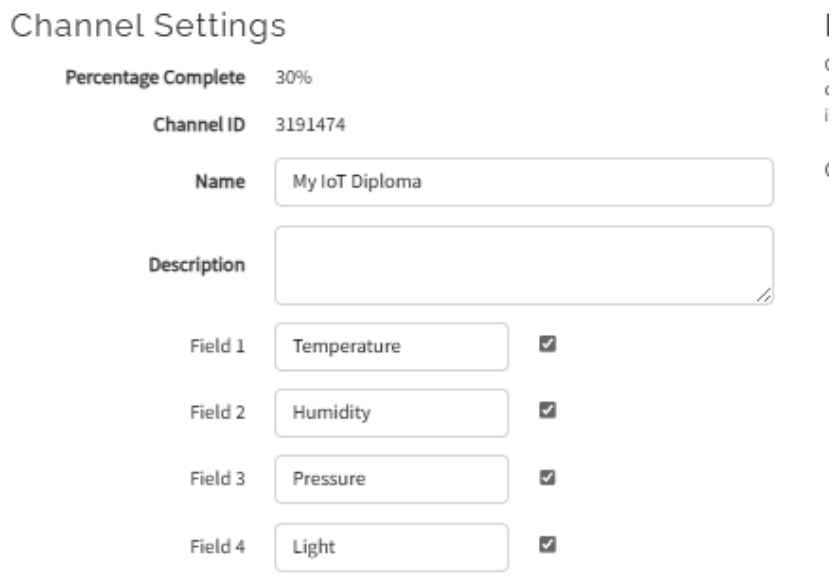


Рисунок 4.7 — Інтерфейс налаштування полів каналу ThingSpeak

На рисунку 4.8 показано інтерфейс керування API-ключами, де відображається ключ запису, ключі читання, а також приклади HTTP-запитів, які можуть використовуватися для тестування або інтеграції з іншими системами. Саме цей ключ включено у прошивку ESP32-S3, що дозволяє пристрою передавати показники сенсорів у хмару без додаткової автентифікації.

Після налаштування каналу користувачу стають доступними додаткові інструменти, зокрема створення візуалізацій та віджетів, що дозволяють аналізувати прийняті дані у веб-інтерфейсі. На рисунку 4.9 наведено відповідний фрагмент меню, у якому можна додати нові графіки, індикатори або здійснити експорт зібраних даних.

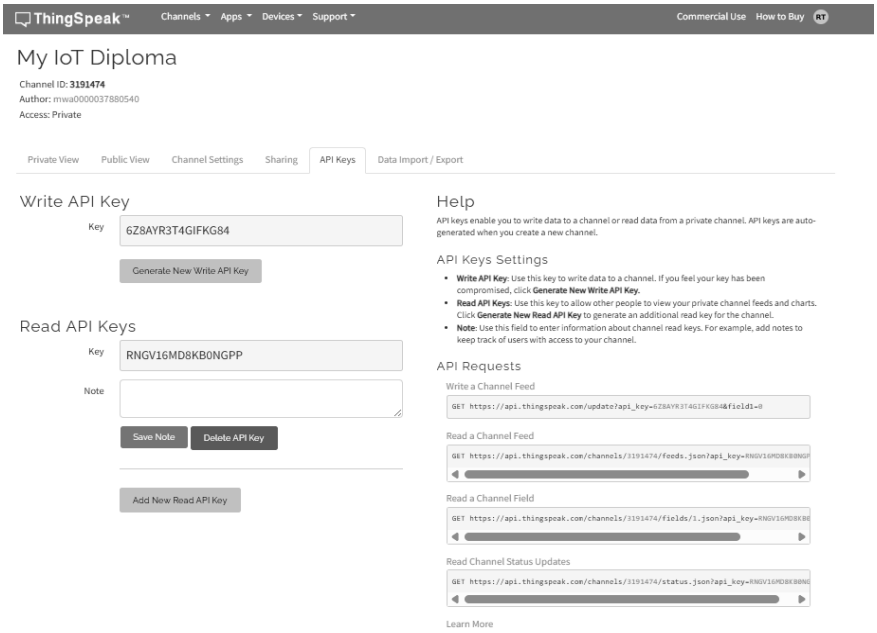


Рисунок 4.8 — Панель керування API-ключами каналу

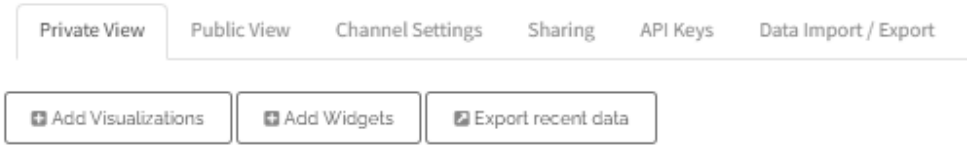


Рисунок 4.9 — Інструменти візуалізації у веб-інтерфейсі ThingSpeak

Для кожного поля ThingSpeak автоматично будує часові графіки після появи перших записів від пристрою. Приклад сформованої веб-панелі моніторингу подано на рисунку 4.10: тут відображені графіки зміни температури, вологості, атмосферного тиску та рівня освітленості у режимі реального часу.

Кожен графік оновлюється автоматично за надходженням нових значень, а зібрані дані можуть зберігатися у необмеженій історичній базі.

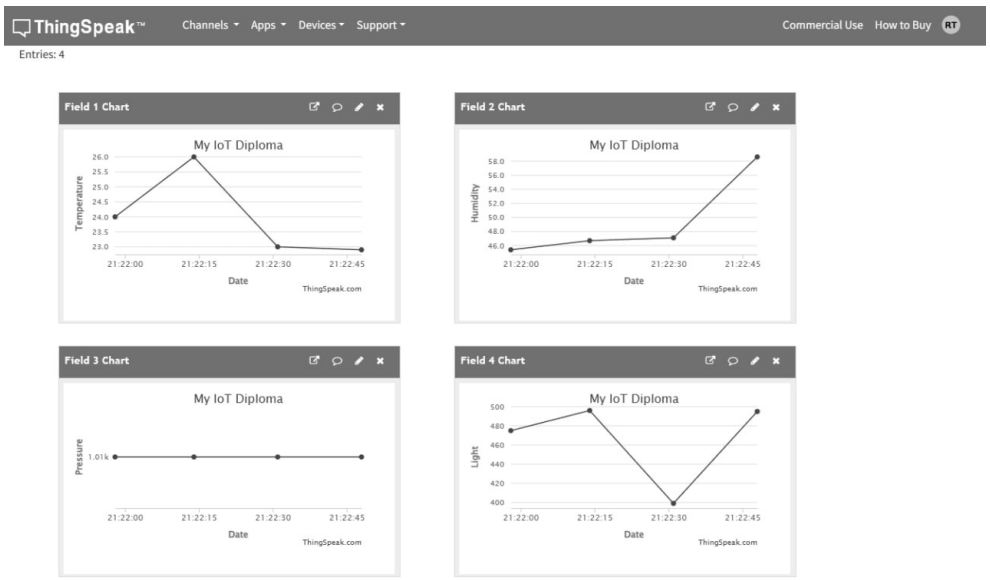


Рисунок 4.10 — Графічна візуалізація телеметрії на хмарній платформі ThingSpeak

Таким чином, платформа ThingSpeak забезпечує повний цикл серверної обробки даних — від приймання телеметрії до візуалізації та подальшої аналітики, що робить її ефективним інструментом для інтеграції з розробленим мікропроцесорним модулем. Блок-схема алгоритму реєстрації пристрою у застосунку показано в Додатку Ж.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Головною ціллю проведення комерційного та технологічного аудиту є розширення функціоналу мікропроцесорного модуля для IoT-застосувань шляхом апаратно-програмного доповнення його засобами вимірювання температури, вологості, освітленості та тиску.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів з ліцею № 1 Крижопільської селищної ради: Козаченко Михайло Борисович, Смішна Оксана Петрівна, Купченко Надія Анатоліївна. Для проведення технологічного аудиту було використано таблицю 5.1 [40], в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 5.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 5.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Козаченко М. Б.	Смішна О. П.	Купченко Н. А.
	Бали, виставлені експертами:		
1	2	2	2
2	3	4	5
3	1	2	2
4	4	3	4

Продовження таблиці 5.3

5	3	4	3
6	1	2	1
7	4	3	3
8	3	4	4
9	1	1	2
10	4	3	4
11	4	4	4
12	3	3	4
Сума балів	$СБ_1$=31	$СБ_2$=34	$СБ_3$=34
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{31+34+34}{3} = 33$		

Середньоарифметична оцінка, отримана на основі експертних висновків, становить 33 бали, і згідно з таблицею 5.2, це вказує на рівень вище середнього комерційного потенціалу результатів проведених досліджень.

Результатом роботи є універсальна багатофункціональна апаратно-програмна платформа , що являє собою готовий апаратно-програмний комплекс для вирішення різних завдань в IoT-застосуваннях

Цільовими користувачами є розробники, які створюють IoT-пристрої різного призначення. Вони отримують готове рішення, що дозволяє суттєво скоротити часові та фінансові витрати на етапі проектування та інтеграції власних продуктів.

В якості аналога для розробки було обрано плату AVR-IoT WA Development Board від компанії Microchip. Основними недоліками аналога є застаріла 8-бітна архітектура (ATmega4808) та двочіпова компоновка. Для реалізації Wi-Fi використовується окремий мережевий контролер (ATWINC1510), що збільшує енергоспоживання та фізичні габарити пристрою. Також до недоліків можна віднести відсутність підтримки Bluetooth LE (BLE), що унеможливлює сучасні методи налаштування (provisioning) та локального керування зі смартфона. Продуктивність 8-бітного МК недостатня для

виконання завдань граничних обчислень (Edge Computing) одночасно з підтримкою захищеного мережевого стека.

У розробці дана проблема вирішується використанням єдиної системи-на-кристалі (SoC) Nano ESP32 (ESP32-S3). Ця платформа має двоядерну 32-бітну архітектуру, що дозволяє (завдяки FreeRTOS) виділити одне ядро під мережеві задачі, а друге — під логіку користувача, забезпечуючи високу продуктивність. Крім того, вона інтегрує Wi-Fi та Bluetooth LE на одному кристалі, що робить пристрій більш компактним, енергоефективним та функціональним.

Існуючі методи вирішення задачі побудови універсального IoT-вузла можна умовно поділити на три основні підходи. Перший — це використання мікроконтролерів (MCU) з низьким ступенем інтеграції, наприклад, 8-бітних AVR або базових STM32, з додаванням зовнішніх модулів (shields) для кожної окремої функції (Wi-Fi, BLE, сенсори). Цей метод є громіздким, енергонеефективним та знижує загальну надійність системи через велику кількість з'єднань.

Другий підхід полягає у використанні одноплатних комп'ютерів (SBC), таких як Raspberry Pi. Він забезпечує високу продуктивність, але є надлишковим для більшості завдань моніторингу, має значне енергоспоживання і не забезпечує детермінізму для операцій в реальному часі.

Третій, найбільш сучасний підхід — це використання високоінтегрованих систем-на-кристалі (SoC), які поєднують МК та радіомодуль на одному чіпі. До цього методу належить як обраний аналог (AVR-IoT), так і наша розробка.

Проект має суттєві переваги над обраним аналогом (AVR-IoT). По-перше, це продуктивність та архітектура: розробка базується на 32-бітній двоядерній архітектурі (ESP32-S3), що дозволяє реалізувати повноцінну багатозадачність (FreeRTOS) , тоді як аналог використовує застаріле 8-бітне одноядерне ядро (ATmega4808). По-друге, це інтеграція: наша розробка використовує єдиний чіп (SoC), де МК та радіомодулі інтегровані; аналог є двочіповим рішенням (ATmega4808 + ATWINC1510), що збільшує енергоспоживання. По-третє, це гібридний зв'язок: розроблений модуль підтримує Wi-Fi та Bluetooth LE (BLE) ,

що є критичним для сучасних методів налаштування, тоді як аналог підтримує лише Wi-Fi. Нарешті, це універсальність та функціонал: наша розробка одразу інтегрує повний набір необхідних сенсорів (BME280, BH1750), RTC (DS3231) та силові ключі (MOSFET), у той час як аналог є "голою" платою, що вимагає купівлі та підключення всіх цих модулів окремо

В таблиці 5.4 наведені основні техніко-економічні показники аналога і нової розробки.

Таблиця 5. 4 – Порівняння аналога з новою розробкою

Показник	Варіанти	
	Базовий (товар-конкурент)	Новий (інноваційне рішення)
1	2	3
1. Нормативно-технічні показники		
Архітектура МК	8-біт / 1-ядро	32-біт / 2-ядра
Наявність Bluetooth LE	-	+
Інтеграція сенсорів	-	BME280, BH1750
Інтеграція RTC	-	DS3231
Наявність силових виходів	-	2x MOSFET
2. Економічні показники		
Ціна придбання, грн	1200	950

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

Основна заробітна плата кожного із дослідників $З_о$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою (5.1):

$$З_0 = \frac{М}{Т_р} * t \text{ (грн)} \quad (5.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Зведемо сумарні розрахунки до таблиця 5.5.

Таблиця 5.5 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	16000	761,9	5	3810
Програміст	12000	571,4	30	17143
Всього				20952

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою (5.2):

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.2)$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою (5.3):

$$C_i = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (5.3)$$

де $М_М$ — розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

$К_і$ — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ — середня кількість робочих днів в місяці, приблизно Тр = 21…23 дні;

$t_{зм}$ — тривалість зміни, год.

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується від 10 до 12 % від основної заробітної плати робітників. Величина витрат на основну заробітну плату робітників наведена в таблиці 5.6.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 11% від основної заробітної плати і визначається формулою (5.4).

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (5.4)$$

$$З_д = 0{,}11 * (20952 + 1033{,}3) = 2418{,}43 \text{ (грн)}$$

Таблиця 5.6 — Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
1.Підготовчі	3	1	47,6	142,9
2.Монтажні	2	3	64,3	128,6
3.Інтеграційні	2	5	81,0	161,9
4.Налагоджувальні	6	2	52,4	314,3
5.Випробувальні	4	4	71,4	285,7
Всього				1033,3

Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.5):

$$Н_{ЗП} = (З_о + З_р + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (5.5)$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників та робітників, грн.;

$З_р$ — основну заробітну плату робітників, грн.;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (20952 + 1033{,}3 + 2418{,}43\ ) * \frac{22}{100} = 5368{,}91 \text{ (грн)}$$

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою (5.6):

$$М = \sum_{i=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{i=1}^{n} В_j \cdot Ц_{вj}, \quad (5.6)$$

де Нj — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

Цj — вартість матеріалу j-го найменування, грн/кг;

Кj — коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

Вj — маса відходів j-го найменування, кг;

Цвj — вартість відходів j-го найменування, грн/кг.

Витрати на комплектуючі вироби *($К_в$)*, які використовують при дослідженні нового технічного рішення, розраховуються, згідно з їхньою номенклатурою, за формулою (5.7).

Проведені розрахунки зведені в таблицю 5.7.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, шт	Вартість витраченого матеріалу, грн
Припой (Sn60Pb40, 1мм)	2200	0,03	66
Флюс (F-5, рідкий)	1500	0,01	15
Спирт ізопропіловий	250	0,05	12,5
Дріт монтажний (МГТФ)	500	0,1	50
З врахуванням коефіцієнта транспортування			157,85

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (5.7)$$

де $Н_j$ — кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ — покупна ціна комплектуючих *j*-го виду, грн;

K_j — коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*).

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень. Для цього користуємось формулою (5.9).

Проведені розрахунки бажано звести до таблиці 5.8.

Таблиця 5.8 — Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Плата Arduino Nano ESP32	1	650	650
Модуль сенсора BME280 (I2C)	1	150	150
Модуль сенсора BH1750 (I2C)	1	80	80
Модуль RTC DS3231 (I2C)	1	100	100
Модуль NFC PN532 (SPI/I2C)	1	350	350
Модуль слоту MicroSD (SPI)	1	40	40
Дисплей OLED 0.96" (I2C)	1	150	150
Мікросхема MCP23017 (Розширювач GPIO)	1	60	60
Мікросхема ATECC608A (Secure Element)	1	400	400
Транзистор MOSFET (IRLZ44N)	2	30	60
Модуль заряду TP4056 (USB-C)	1	35	35
Стабілізатор LDO (ADP3338 / AMS1117-3.3)	1	25	25
Пасивні компоненти та роз'єми	1	200	200
Всього з врахуванням транспортних витрат			2530,00

Таблиця 5.9 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп'ютер	30000	2	1	1250,00
Паяльна станція	1500	4	1	31,25
Всього				1281,25

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_і} \quad (5.9)$$

де W_{yt} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$\eta_і$ — коефіцієнт корисної дії обладнання, $\eta_і < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}5 \cdot 200 \cdot 12{,}69 \cdot 0{,}5}{0{,}8} = 793{,}13$$

Витрати за статтею «Службові відрядження» розраховуються від 20 до25% від суми основної заробітної плати дослідників та робітників за формулою (5.10):

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%}, \quad (5.10)$$

де $Н_{св}$ — норма нарахування за статтею «Службові відрядження».

$$В_{св} = 0{,}2 * (20952 + 1033{,}3) = 4397{,}14$$

Накладні (загальновиробничі) витрати $В_{нзв}$ охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати

$В_{нзв}$ можна прийняти як (від 100 до 150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто за формулою (5.11):

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.11)$$

де $Н_{нзв}$ — норма нарахування за статтею «Інші витрати».

$$В_{нзв} = (20952 + 1033{,}3) \cdot \frac{100}{100\%} = 21985{,}71 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

$$В=20952+1033{,}3+2418{,}43+5368{,}91+157{,}85+2530+1281{,}25+793{,}13+4397{,}14+21985{,}71= 60918{,}14 \text{грн}$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою (5.12):

$$ЗВ = \frac{В}{\eta}, \quad (5.12)$$

де η — коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт $\beta = 0{,}7$.

Звідси:

$$ЗВ = \frac{60918{,}14}{0{,}7} = 87025{,}9 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна

отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою (5.13):

$$\Delta П_i = \sum_{1}^{n} \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N \right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100} \right) \quad (5.13)$$

де $\Delta Ц_0$ — покращення основного оціночного показника від впровадження результатів розробки у даному році.

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

л — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333;

р — коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х — ставка податку на прибуток. У 2025 році — 18%.

Припустимо, що ціна зросте на 100 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 500 шт., протягом другого року — на 600 шт., протягом третього року на 700 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до 950 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [100 \cdot 1 + (950 + 100) \cdot 500] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 89700{,}99 \text{ грн.}$$

$$\Delta П_2 = [100 \cdot 1 + (950 + 100) \cdot (500 + 600)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 197404{,}61 \text{ грн.}$$

$$\Delta П_3 = [100 \cdot 1 + (950 + 100) \cdot (500 + 600 + 700)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 322962{,}09 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки за формулою (5.14):

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \tag{5.14}$$

де $k_{\text{інв}}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію.

Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$ = від 2 до 5).

$$PV = 2 \cdot 87025{,}9 = 174051{,}82$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули (5.15):

$$E_{абс} = (ПП - PV) \tag{5.15}$$

де ПП — приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн. яка визначається формулою (5.16):

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{5.16}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$\text{ПП} = \frac{89700{,}99}{(1+0{,}2)^1} + \frac{197404{,}61}{(1+0{,}2)^2} + \frac{322962{,}09}{(1+0{,}2)^3} = 399606{,}01\text{грн.}$$

$$\text{Е}_{\text{абс}} = (399606{,}01 - 174051{,}82) = 225554{,}19 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою (5.17):

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.17}$$

де $T_ж$ – життєвий цикл наукової розробки, роки.

$$\text{Е}_\text{в} = \sqrt[3]{1 + \frac{225554{,}19}{174051{,}82}} - 1 = 0{,}53 = 53\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою (5.18):

$$\tau = d + f, \tag{5.18}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d= (від 0,14 до 0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (від 0,05 до 0,1).

$$\tau_{\min} = 0{,}18 + 0{,}05 = 0{,}23.$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою (5.19):

$$T_{ок} = \frac{1}{E_в} \tag{5.19}$$

$$\text{Т}_\text{ок} = \frac{1}{0{,}53} = 1{,}9 \text{ роки}$$

Так як $T_{ок} = 1{,}9$ роки, то фінансування даної наукової розробки в принципі є доцільним.

ВИСНОВКИ

Технологія Інтернет речей є однією з таких, що бурно розвивається, оскільки надає можливість дистанційного контролю та управління. Аналіз сучасного стану та проблем розвитку Інтернету речей показав, що фрагментованість стандартів, підвищені вимоги до безпеки та енергоефективності зумовлюють необхідність створення універсальних мікропроцесорних модулів для IoT-застосувань.

Проведений огляд мікроконтролерних платформ для використання як бази для розробки IoT-пристроїв показав, що існуючі рішення є або вузькоспеціалізованими, або орієнтованими на закриті програмні системи, вимагають застосування додаткових елементів для реалізації кінцевого продукту.

На основі апаратної платформи ESP32-S3 (Arduino Nano ESP32), яка забезпечує оптимальний баланс між обчислювальною потужністю, функціональними можливостями, що надаються, та вартістю, було розроблено універсальний мікропроцесорний засіб, що інтегрує високопродуктивний двоядерний мікроконтролер з компонентам підтримки бездротового зв’язку Wi-Fi та Bluetooth, сенсори вологості, температури, тиску та освітленості, підсистему автономної роботи та засоби управління електричним навантаженням.

Розроблений універсальний мікропроцесорний модуль є готовим рішенням, що може бути використане як базова апаратно-програмна платформа для побудови різноманітних пристроїв для застосування у широкому спектрі IoT-застосувань без необхідності додаткового апаратного доопрацювання.

Програмне забезпечення модуля побудовано з використанням операційної системи реального часу FreeRTOS, що забезпечує багатозадачну обробку даних, розподіл функцій між ядрами мікроконтролера та детерміновану роботу системи в умовах активної мережевої взаємодії.

Оцінка комерційного потенціалу розробки показала, що вкладені в даний проект інвестиції окупляться через 1,9 роки. Загальні витрати складають 87025,9 грн. Прибуток за три роки складає 399606,01 тис. грн.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ткаченко Р.В., Тарновський М.Г. Універсальний мікропроцесорний засіб для IoT застосувань Конференція ВНТУ: Молодь в науці: дослідження, проблеми, перспективи (МН-2026). Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26630

2. Sharma S., Bhatt C. A Comprehensive Review on IoT: Architectures, Key Components, and Future Trends // International Journal of Computer Science and Network Security. 2023. Vol. 23, No. 7. P. 1–15.

3. Gubbi J., Buyya R., Marusic S., Palaniswami M. Internet of Things (IoT): A vision, architectural elements, and future directions // Future Generation Computer Systems. 2013. Vol. 29, No. 7. P. 1645–1660. DOI: 10.1016/j.future.2013.01.010.

4. Mainetti L., Patrono L., Stefanizzi M. An IoT-Aware Architecture for Smart Building Monitoring // Sensors. 2022. Vol. 22, No. 14. P. 5369. [Електронний ресурс]. Режим доступу: https://www.mdpi.com/1424-8220/22/14/5369 (дата звернення: 06.11.2025).

5. Hoang A. T. Fog Computing for IoT // The Next Generation of Internet of Things (IoT) / ed. by Tyagi A. K., Abraham A., Balas V. E. Cham: Springer International Publishing, 2022. P. 241–264.

6. Hannan M. F., Ali J. A., Ker P. J. Machine-to-Machine Communication for IoT // Encyclopedia of Wireless Networks / ed. by Shen S., Lin X., Zhang K. Cham: Springer International Publishing, 2023. P. 1–9.

7. Nguyen C. P. T., Hoang D. T., Nguyen M. N. Security and Privacy for IoT: A Survey of Methodologies, Solutions, and Challenges // IEEE Internet of Things Journal. 2022. Vol. 9, No. 1. P. 158–197.

8. Arduino. Arduino Nano ESP32 Documentation. [Електронний ресурс]. Режим доступу: https://docs.arduino.cc/hardware/nano-esp32 (дата звернення: 06.11.2025).

9. Espressif Systems. ESP32 Series Datasheet : Technical documentation. 2022. [Електронний ресурс]. Режим доступу:

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 06.11.2025).

10. Espressif Systems. ESP32-S3 Series Datasheet : Technical documentation. [Електронний ресурс]. Режим доступу: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf (дата звернення: 06.11.2025).

11. Microchip Technology Inc. AVR-IoT WA Development Board. [Електронний ресурс]. Режим доступу: https://www.microchip.com/en-us/development-tool/ev15r70a (дата звернення: 06.11.2025).

12. Microchip Technology Inc. ATmega4808/4809 Datasheet : Technical documentation. 2019. [Електронний ресурс]. Режим доступу: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega4808-4809-Data-Sheet-DS40002173B.pdf (дата звернення: 06.11.2025).

13. M5Stack. M5Stack CoreS3 ESP32-S3 IoT Development Kit. [Електронний ресурс]. Режим доступу: https://shop.m5stack.com/products/m5stack-cores3-esp32-s3-iot-development-kit (дата звернення: 06.11.2025).

14. Adafruit Industries. Adafruit ESP32-S3 Feather with STEMMA QT. [Електронний ресурс]. Режим доступу: https://www.adafruit.com/product/5303 (дата звернення: 06.11.2025).

15. Particle. Particle Argon Wi-Fi + Mesh Development Kit. [Електронний ресурс]. Режим доступу: https://store.particle.io/products/argon (дата звернення: 06.11.2025).

16. IoT Mill – Prototyping Platform. [Електронний ресурс]. Режим доступу: https://iotmill.com/ (дата звернення: 06.11.2025).

17. Li S., Wang C., Zhang W. A Survey on Wi-Fi for IoT: Challenges and Opportunities // IEEE Internet of Things Journal. 2023. Vol. 10, No. 15. P. 13349–13370.

18. Gomez C., Oller J., Paradells J. Overview and Evaluation of Bluetooth Low Energy // Sensors. 2012. Vol. 12, No. 9. P. 11734–11753.

19. Al-Masri H. J. MQTT protocol for IoT applications: A survey // International Conference on IoT, Communication, and Technology (ICICT). 2023. P. 1–6.

20. Leens F. An introduction to I2C and SPI protocols // IEEE Instrumentation & Measurement Magazine. 2009. Vol. 12, No. 1. P. 8–13. DOI: 10.1109/MIM.2009.4762946.

21. NXP Semiconductors. UM10204: I2C-bus specification and user manual. Rev. 7.0. 2021. [Електронний ресурс]. Режим доступу: https://www.nxp.com/docs/en/user-guide/UM10204.pdf (дата звернення: 06.11.2025).

22. Valvano J. W. Embedded Systems: Real-Time Interfacing to ARM Cortex-M Microcontrollers. 5th ed. CreateSpace Independent Publishing Platform, 2015. 642 p.

23. Barry R. Mastering the FreeRTOS Real Time Kernel: A Hands-On Tutorial Guide. Real Time Engineers Ltd, 2018. 450 p.

24. Espressif Systems. ESP-IDF Programming Guide. [Електронний ресурс]. Режим доступу: https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/index.html (дата звернення: 06.11.2025).

25. Pathak A., Motamedi S. A Review of Energy-Efficient Techniques in IoT Systems // Journal of Low Power Electronics and Applications. 2022. Vol. 12, No. 3. P. 34.

26. Noergaard T. Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers. 2nd ed. Elsevier Science, 2018. 576 p.

27. Volos C. K., Kyprianidis I. M. (eds). Microcontroller-Based Systems: Design, Implementation, and Applications. MDPI, 2023. 432 p. [Електронний ресурс]. Режим доступу: https://www.mdpi.com/books/book/7520 (дата звернення: 06.11.2025).

28. Mordor Intelligence. Global IoT Market: Growth, Trends, COVID-19 Impact, and Forecasts (2023–2028). [Електронний ресурс]. Режим доступу:

https://www.mordorintelligence.com/industry-reports/internet-of-things-iot-market (дата звернення: 07.11.2025).

29. Bosch Sensortec. BST-BME280-DS002 Datasheet : Technical documentation. [Електронний ресурс]. Режим доступу: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf (дата звернення: 07.11.2025).

30. Rohm Semiconductor. BH1750FVI-E Datasheet. [Електронний ресурс]. Режим доступу: https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf (дата звернення: 07.11.2025).

31. Analog Devices. DS3231 Datasheet: Extremely Accurate I²C-Integrated RTC/TCXO/Crystal. [Електронний ресурс]. Режим доступу: https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231.pdf (дата звернення: 07.11.2025).

32. Arduino. Arduino and SD Cards. [Електронний ресурс]. Режим доступу: https://docs.arduino.cc/learn/programming/sd-guide (дата звернення: 07.11.2025).

33. Boylestad R. L., Nashelsky L. Electronic Devices and Circuit Theory. 11th ed. Pearson, 2018. 1168 p.

34. Coskun V., Ozdenizci B., Ok K. A survey on near field communication (NFC) technology // Wireless Personal Communications. 2013. Vol. 71, No. 3. P. 2259–2294.

35. Buchmann J. Battery Management Systems for Large Lithium-Ion Battery Packs. 1st ed. Apress, 2016. 233 p.

36. Basso C. Switch-Mode Power Supplies: SPICE Simulations and Practical Designs. 2nd ed. McGraw-Hill, 2021. 1008 p.

37. Analog Devices. Linear Circuit Design Handbook. [Електронний ресурс]. Режим доступу: https://www.analog.com/en/education/education-library/linear-circuit-design-handbook.html (дата звернення: 07.11.2025).

38. Microchip Technology Inc. ATECC608A TrustFLEX: Secure Element. [Електронний ресурс]. Режим доступу: https://www.microchip.com/en-us/product/ATECC608A (дата звернення: 07.11.2025).

39. Захарченко С. М., Рябенький В. М., Усатюк В. М. Інформаційні технології та протоколи в системах Інтернету речей: навч. посіб. Вінниця: ВНТУ, 2021. 148 с.

40. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н. О.Д. Азаров

«03» жовтня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

універсальний мікропроцесорний модуль для IoT застосувань

Керівник роботи: к.т.н., доц. каф. ОТ:

_____Тарновський М.Г.

Виконав: студент гр. 1КІ-24м

_____Ткаченко Р.В.

2025

1 Підстава для виконання бакалаврського дипломного проекту (МКР)

1.1 Актуальність зумовлена зростаючою популярністю IoT-застосувань, що створює попит на універсальні багатофункціональні апаратно-програмні платформи, використання яких дозволяє спростити розробку та впровадження кінцевого продукту IoT;

1.2 Наказ про затвердження теми МКР № 313 від 24.09.2024 р.

2 Мета і призначення МКР

2.1 Метою проекту є розширення функціональних можливостей мікропроцесорного модуля для IoT застосувань за рахунок апаратно-програмної інтеграції у ньому засобів для вимірювання температури, вологості, освітленості та тиску, а також забезпечення автономності та захищеної передачі даних;

2.2 Призначення розробки — створення універсальної апаратно-програмної платформи як основи для розробки пристроїв, що можуть бути інтегровані у різні IoT-системи.

3 Вихідні дані для виконання МКР

3.1 Призначення — апаратно-програмна платформа для розробки пристроїв IoT.

3.2 Інтерфейс підключення до мережі Інтернет — WiFi.

3.3 Кількість підтримуваних сенсорів — не менше 3.

3.4 Кількість каналів управління — не менше 2.

3.5 Живлення — від зовнішнього джерела постійної напруги та від акумулятора.

3.6 Технічна документація на мікроконтролер ESP32-S3, специфікації сенсорних модулів BME280, BH1750 та модуля реального часу DS3231.

3.7 Програмне середовище ESP-IDF, бібліотеки FreeRTOS.

4 Технічні вимоги до виконання МКР

4.1 Провести аналіз апаратних і програмних засобів для побудови універсальних IoT-модулів.

4.2 Провести аналіз теоретичних основи архітектур та протоколів IoT-систем.

4.3 Вибрати мікропроцесорну платформу та основні компоненти пристрою.

4.4 Розробити структурну та функціональну схеми модуля.

4.5 Розробити програмне забезпечення у середовищі FreeRTOS із розподілом задач між потоками.

4.6 Розробити рекомендації з введення в експлуатацію та налаштування.

4.7 Оцінити комерційний потенціал розробки.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1

Таблиця А.1 — Етапи МКР

№ з/п	Назва етапів дипломної Роботи	Термін виконання		Очікувані результати
		початок	закінчення	
1	Постановка задачі роботи	08.09.2025	08.09.2025	Задачі дослідження
2	Обгрунтування актуальності теми. Аналіз сучасних рішень	08.09.2025	20.09.2025	Розділ 1
3	Теоретичне обґрунтування архітектури та протоколів	20.09.2025	27.09.2025	Розділ 2
4	Розробка апаратної частини та програмного забезпечення	27.09.2025	30.09.2025	Розділ 3
5	Налагодження, тестування та інтеграція з хмарою	30.09.2025	15.10.2025	Розділ 4
6	Розрахунок економічної ефективності	15.10.2025	17.10.2025	Розділ 5
7	Оформлення матеріалів до захисту МКР	17.10.2025	04.11.2025	ПЗ, графічний матеріал і презентація

Продовження таблиці А.1

8	Перевірка якості виконання магістерської роботи та усунення недоліків	04.11.2025	10.11.2025	Оформлені документи
9	Підписи супроводжувальних документів у нормоконтролера, керівника, опонента			Оформлені документи
10	Перевірка на антиплагіат			Оформлені документи

6 Матеріали, що подаються до захисту МКР

Матеріали, що подаються до захисту МКР: пояснювальна записка МКР, графічні та ілюстративні матеріали, протокол попереднього захисту роботи на кафедрі, відзив наукового керівника, рецензія рецензента, анотації до МДП українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21.

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Універсальний мікропроцесорний модуль для IoT застосувань

Тип роботи: магістерська кваліфікаційна робота

(бакалаврська кваліфікаційна робота/магістерська кваліфікаційна робота)

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 16 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Азаров О. Д., д.т.н., зав. каф. ОТ _____
(прізвище, ініціали, посада) (підпис)

Мартинюк Т. Б., д.т.н., проф. каф. ОТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С. М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Тарновський М. Г., доц. каф. ОТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Ткаченко Р. В.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Структурна схема універсального мікропроцесорного модуля

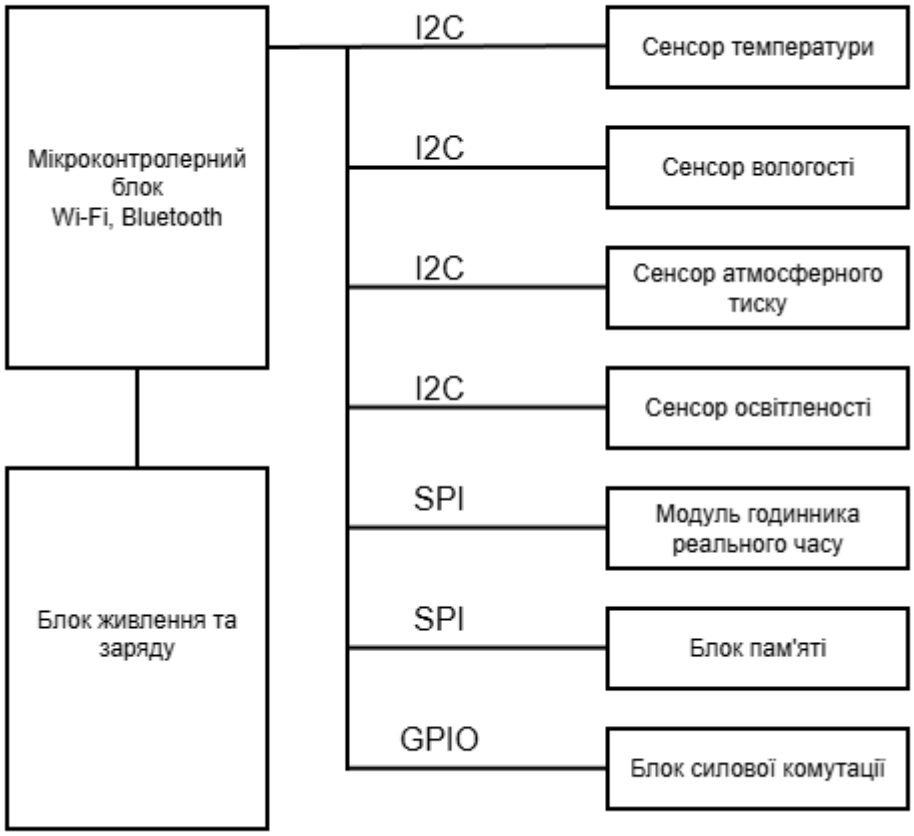


Рисунок В.1 — Структурна схема універсального мікропроцесорного модуля

ДОДАТОК Г

Функціональна схема універсального мікропроцесорного модуля

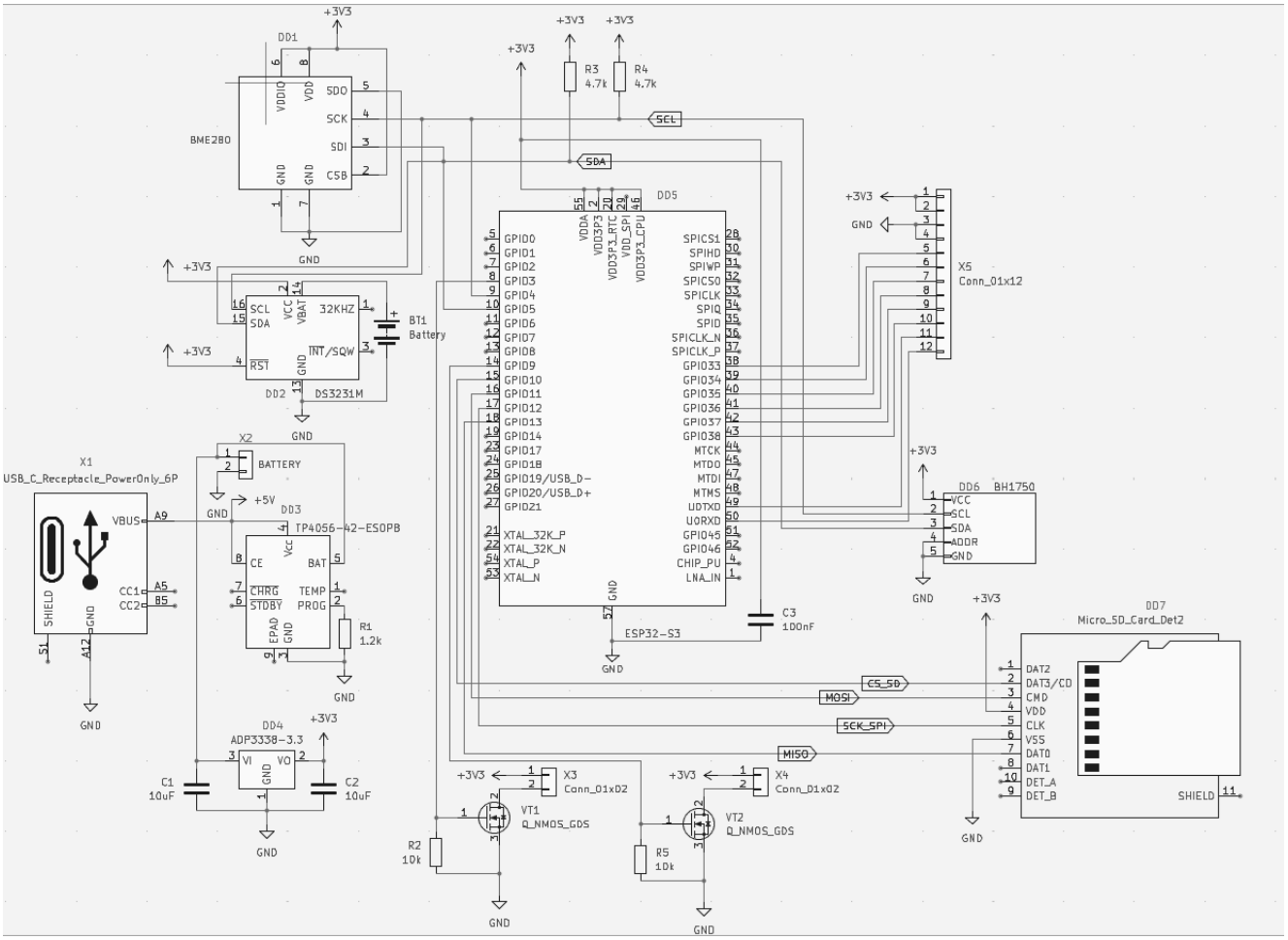


Рисунок Г.1 — Функціональна схема універсального мікропроцесорного модуля

ДОДАТОК Д

Блок-схема програмного забезпечення універсального мікропроцесорного модуля

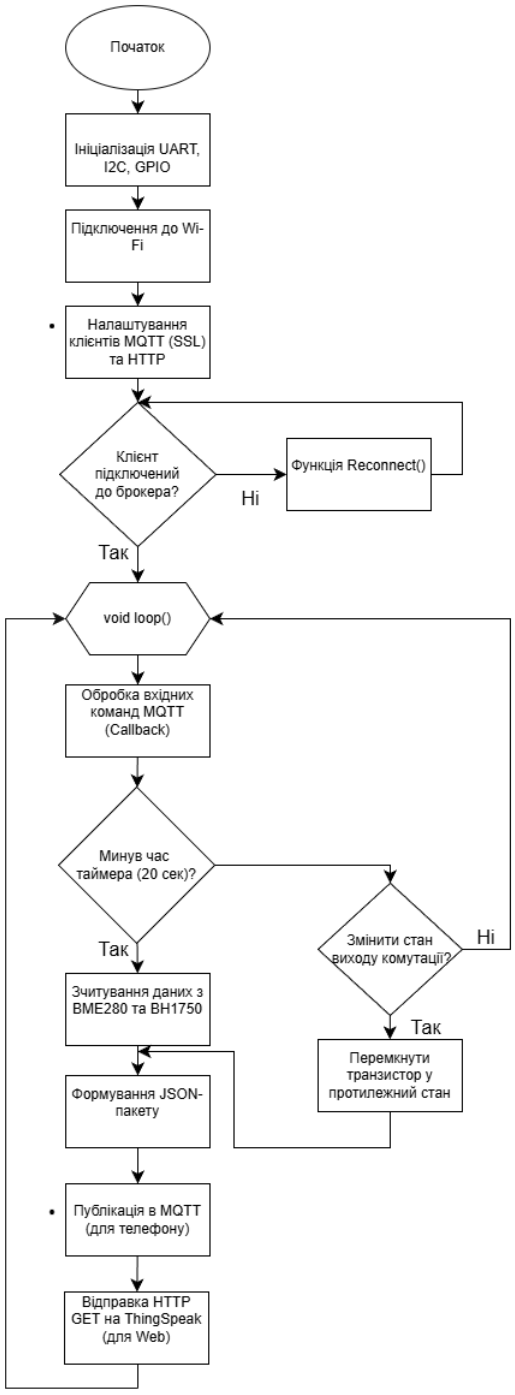


Рисунок Д.1 — Блок-схема програмного забезпечення універсального мікропроцесорного модуля

ДОДАТОК Е

Лістинг програмного забезпечення універсального мікропроцесорного модуля

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <HTTPClient.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <BH1750.h>
#include <RTClib.h>
#include <SPI.h>
#include <SD.h>
#include <ArduinoJson.h>
const char* ssid     = "YOUR_WIFI_NAME";
const char* password = "YOUR_WIFI_PASSWORD";
const char* mqtt_server = "broker.hivemq.com";
const int   mqtt_port   = 1883;
const char* topic_data  = "device/nano_esp32_unique_id/sensors";
const char* topic_ctrl1 = "device/nano_esp32_unique_id/control1";
const char* topic_ctrl2 = "device/nano_esp32_unique_id/control2";
const char* thingspeak_api_key = "YOUR_WRITE_API_KEY";
const char* thingspeak_server  = "https://api.thingspeak.com/update";
#define SDA_PIN     9
#define SCL_PIN     10
#define MOSFET1_PIN 21
#define MOSFET2_PIN 4
#define SD_MOSI 35
#define SD_MISO 37
#define SD_SCK  36
#define SD_CS   34
#ifndef LED_BUILTIN
#define LED_BUILTIN 13
#endif
WiFiClient espClient;
PubSubClient client(espClient);
Adafruit_BME280 bme;
BH1750 lightMeter;
RTC_DS3231 rtc;
bool bme_ok = false;
bool bh_ok  = false;
bool rtc_ok = false;
bool sd_ok  = false;
unsigned long lastMsg = 0;
const unsigned long interval = 5000;
void callback(char* topic, byte* payload, unsigned int length) {
  String msg;
  for (unsigned int i = 0; i < length; i++) {
    msg += (char)payload[i];
```

```
  }
  if (String(topic) == topic_ctrl1) {
    digitalWrite(MOSFET1_PIN, (msg == "ON") ? HIGH : LOW);
  }
  else if (String(topic) == topic_ctrl2) {
    digitalWrite(MOSFET2_PIN, (msg == "ON") ? HIGH : LOW);
  }}
void reconnect() {
  while (!client.connected()) {
    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      client.subscribe(topic_ctrl1);
      client.subscribe(topic_ctrl2);
    } else {
      delay(5000);
    }
  }
}
void sendToThingSpeak(float temp, float hum, float pres, float lux) {
  if (WiFi.status() != WL_CONNECTED) return;
  HTTPClient http;
  String url = String(thingspeak_server) +
               "?api_key=" + thingspeak_api_key +
               "&field1=" + String(temp, 2) +
               "&field2=" + String(hum, 2) +
               "&field3=" + String(pres, 2) +
               "&field4=" + String(lux, 2);
  http.begin(url);
  http.GET();
  http.end();
}
void setup() {
  Serial.begin(115200);
  pinMode(MOSFET1_PIN, OUTPUT);
  pinMode(MOSFET2_PIN, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(MOSFET1_PIN, LOW);
  digitalWrite(MOSFET2_PIN, LOW);
  digitalWrite(LED_BUILTIN, LOW);
  Wire.begin(SDA_PIN, SCL_PIN);
  if (bme.begin(0x76)) bme_ok = true;
  if (lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE)) bh_ok = true;
  if (rtc.begin()) rtc_ok = true;
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  SPI.begin(SD_SCK, SD_MISO, SD_MOSI);
  sd_ok = SD.begin(SD_CS);
  client.setServer(mqtt_server, mqtt_port);
```

```
  client.setCallback(callback);
}
void loop() {
  if (!client.connected()) reconnect();
  client.loop();
  unsigned long now = millis();
  if (now - lastMsg > interval) {
    lastMsg = now;
    float temp = 0, hum = 0, pres = 0, lux = 0;
    if (bme_ok) {
      temp = bme.readTemperature();
      hum  = bme.readHumidity();
      pres = bme.readPressure() / 100.0F;
    }
    if (bh_ok) {
      lux = lightMeter.readLightLevel();
    }
    char ts[25] = "";
    if (rtc_ok) {
      DateTime t = rtc.now();
      snprintf(ts, sizeof(ts), "%04d-%02d-%02dT%02d:%02d:%02d",
               t.year(), t.month(), t.day(),
               t.hour(), t.minute(), t.second());
    }
    DynamicJsonDocument doc(256);
    doc["temp"]     = temp;
    doc["humidity"] = hum;
    doc["pressure"] = pres;
    doc["lux"]      = lux;
    if (rtc_ok) doc["ts"] = ts;
    String payload;
    serializeJson(doc, payload);
    client.publish(topic_data, payload.c_str());
    sendToThingSpeak(temp, hum, pres, lux);
    if (sd_ok) {
      File f = SD.open("/log.txt", FILE_APPEND);
      if (f) {
        f.println(payload);
        f.close();
      }
    }
  }}
```

ДОДАТОК Ж

Блок-схема алгоритму реєстрації універсального мікропроцесорного модуля у застосунку

Рисунок Е.1 — Блок-схема алгоритму реєстрації універсального мікропроцесорного модуля у застосунку