Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет __Кібербезпеки_____
(повна назва)

Кафедра _Інформаційно-мережної інженерії_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____
_Оптимізація передачі даних у бездротових мережах для дистанційного_____
керування обладнанням на базі Arduino_____
(тема)

Виконав:
здобувач ___4__ року навчання,
групи __ТРІМІ-22-1_____
_Олег ХАРЧЕНКО_____
(власне ім'я, прізвище)

Спеціальність _172 Телекомунікації та радіотехніка_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна____
Освітня програма _Інформаційно-мережна інженерія_____
(повна назва освітньої програми)

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

Допускається до захисту

В. о. завідувача кафедри _____ _Дарія ЧЕБОТАРЬОВА__
(підпис) (власне ім'я, прізвище)

2026 р.

Не містить відомостей заборонених до відкритого публікування.

Студент / Олег ХАРЧЕНКО /

Керівник / Галина ЛЯШЕНКО /

Харківський національний університет радіоелектроніки

Факультет Кібербезпеки

Кафедра Інформаційно-мережної інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

В.о. зав. кафедри _____
(підпис)
«____»_____ 20 26 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Харченку Олегу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Оптимізація передачі даних у бездротових мережах для дистанційного керування обладнанням на базі Arduino

затверджена наказом університету від 02 червня 2026 р. № 466 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2026р.

3. Вихідні дані до роботи *Дослідити можливості платформи Arduino як засобу автоматизації, порівняти бездротові модулі передачі даних, розробити алгоритм компенсації інерційності вимірювального середовища. Дослідити процес передачі даних.*

4. Перелік питань, що потрібно опрацювати в роботі

1. Теоретичні основи бездротових систем керування на базі Arduino
2. Аналіз існуючих рішень та підходів
3. Моделювання автоматизованої системи поливу
4. Реалізація та тестування системи

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання 02 червня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Галина ЛЯШЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 86 с., 15 рис., 8 табл., 3 дод., 43 джерела.

ОПТИМІЗАЦІЯ, БЕЗДРОТОВІ МЕРЕЖІ, ІНТЕРНЕТ РЕЧЕЙ, МІКРОКОНТРОЛЕР, ARDUINO, MQTT, ДИСТАНЦІЙНЕ КЕРУВАННЯ, ЕНЕРГОЕФЕКТИВНІСТЬ.

Об’єкт дослідження – процес функціонування бездротових каналів зв'язку в інфраструктурі керуючих систем, побудованих на базі мікроконтролерних платформ сімейства Arduino.

Предмет дослідження – методи та алгоритми оптимізації передачі даних для підвищення надійності та енергоефективності.

Мета роботи – вивчення сучасних методів оптимізації передачі даних у бездротових мережах і створення апаратно-програмної системи дистанційного керування обладнанням на базі Arduino, адаптованої для роботи в умовах нестабільного зв’язку.

У роботі розглядаються поточні бездротові технології та проблеми передачі даних у системах Інтернету речей. Спроєктовано та розроблено апаратно-програмний макет системи автоматизованого поливу для фільтрації шуму датчиків за допомогою Edge Computing. Розроблено подійно-орієнтований алгоритм відправки даних, який використовує бінарне пакування телеметрії за допомогою протоколу MQTT. Проведено експериментальне тестування, а також проведено аналіз ефективності запропонованих рішень.

Системи та алгоритми, які були розроблені, можуть бути використані для створення надійних рішень Інтернету речей (IoT), систем сільськогосподарської автоматизації та програмування мікроконтролерів.

ABSTRACT

Explanatory note of the qualification work: 86 p., 15 pt., 8 tabl., 3 app., 43 sources.

OPTIMIZATION, WIRELESS NETWORKS, INTERNET OF THINGS, MICROCONTROLLER, ARDUINO, MQTT, REMOTE CONTROL, ENERGY EFFICIENCY.

The object of the study is the operation of wireless communication channels in the control system infrastructure based on the Arduino microcontroller platforms.

The subject of the study is methods and algorithms for optimizing data transmission to increase reliability and energy efficiency.

The purpose of the work is to study modern methods for optimizing data transmission in wireless networks and create a hardware-software system for remote control of equipment based on Arduino, adapted to work in unstable communication conditions.

The work considers current wireless technologies and data transmission problems in Internet of Things systems. A hardware and software mock-up of an automated irrigation system for filtering sensor noise using Edge Computing has been designed and developed. An event-driven data sending algorithm has been developed that uses binary telemetry packaging using the MQTT protocol. Experimental testing has been conducted, and the effectiveness of the proposed solutions has been analyzed.

The systems and algorithms that have been developed can be used to create reliable Internet of Things (IoT) solutions, agricultural automation systems, and microcontroller programming.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 10
ВСТУП 13
1 ТЕОРЕТИЧНІ ОСНОВИ БЕЗДРОТОВИХ СИСТЕМ КЕРУВАННЯ НА БАЗІ ARDUINO 14
1.1 Arduino як платформа для систем керування 14
1.1.1 Апаратна частина плат Arduino та її функціонування 15
1.1.2 Arduino IDE як програмний сегмент Arduino 20
1.1.3 Мови програмування для плат Arduino 21
1.2 Технології бездротових мереж 22
1.2.1 Wi-Fi 23
1.2.2 Bluetooth 24
1.2.3 LoRa 25
1.2.4 Zigbee 25
1.2.5 Z-Wave 26
1.2.6 RF 27
1.2.7 InfraRed 27
1.2.8 GSM/LTE 28
1.2.9 Порівняльний аналіз бездротових інтерфейсів за критеріями IoT 29
1.3 Виклики оптимізації передачі даних у бездротових мережах 31
1.3.1 Затримка 31
1.3.2 Надійність 31
1.3.3 Енергоефективність 32
1.3.4 Пропускна спроможність 32
1.3.5 Безпека 33
1.3.6 Масштабованість 33
1.4 Протоколи прикладного рівня для систем дистанційного керування 34
1.4.1 Протокол MQTT та механізми керування якістю обслуговування (QoS) 34
1.4.2 Протокол CoAP 35
1.4.3 Специфіка побудови користувацьких бінарних протоколів 36

1.5 Методи та алгоритми оптимізації інформаційного обміну 37
1.5.1 Формати представлення та структуризації даних у системах з обмеженими ресурсами 37
1.5.2 Подійно-орієнтовані алгоритми відправки даних 38
1.5.3 Методи фільтрації та попередньої обробки даних на межі мережі ... 38
1.6 Висновки розділу 1 39
2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПІДХОДІВ 40
2.1 Комерційні рішення для IoT систем керування 40
2.1.1 Платформи з готовими хмарними сервісами (AWS IoT, Blynk, Arduino Cloud) 40
2.1.2 Відкриті рішення для локального керування (Home Assistant, Node-RED) 41
2.2 Аналіз наукових досліджень щодо оптимізації передачі даних в IoT 42
2.2.1 Оптимізація енергоспоживання та обсягу даних 42
2.2.2 Підвищення надійності доставки (QoS) 43
2.2.3 Фільтрація даних на межі мережі (Edge Computing) 44
2.3 Порівняльний аналіз та постановка задачі 44
2.4 Висновки розділу 2 46
3 ПРОЄКТУВАННЯ СИСТЕМИ ОПТИМІЗАЦІЇ 48
3.1 Обґрунтування вибору технологічного стеку та компонентів 48
3.1.1 Апаратна база системи поливу 48
3.1.2 Технологія зв'язку та протокол прикладного рівня 51
3.1.3 Мова програмування та IDE 52
3.2 Апаратна архітектура та схема з'єднань 53
3.2.1 Структурна схема системи 53
3.2.2 Електрична схема підключення 54
3.3 Програмна архітектура та алгоритм роботи 54
3.3.1 Блок-схема роботи контролера 54
3.3.2 Структура пакету даних 56
3.4 Висновки розділу 3 56
4 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА 57
4.1 Розробка програмного забезпечення системи 57

4.1.1 Реалізація зчитування даних та керування клапаном 57
4.1.2 Реалізація мережевої взаємодії та оптимізації 58
4.1.3 Серверна частина (приймач) 59
4.2 Експериментальне тестування системи 61
4.2.1 Опис тестового стенду 61
4.2.2 Методика тестування оптимізації 62
4.3 Результати експериментів 63
4.3.1 Аналіз обсягу трафіку та навантаження на мережу 63
4.3.2 Надійність та затримки керування 64
4.4 Аналіз результатів та рекомендації 65
4.4.1 Вирішення виявлених проблем 65
4.4.2 Можливості масштабування 65
4.5 Висновки розділу 4 66
ВИСНОВКИ 67
ПЕРЕЛІК ПОСИЛАНЬ 69
ДОДАТОК А СЛАЙДИ ПРЕЗЕНТАЦІЇ 75
ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ ARDUINO 79
ДОДАТОК В ЛІСТИНГ ПРОГРАМИ PYTHON ДЛЯ ОТРИМАННЯ ІНФОРМАЦІЇ ЩОДО СТАНУ ПОЛИВУ ТА ПАКЕТІВ MQTT 84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AES (Advanced Encryption Standard) – розширений стандарт шифрування

ASCII (American Standard Code for Information Interchange) – американський стандартний код для обміну інформацією

ATT (Attribute Protocol) – протокол атрибутів, який використовується в технології BLE

AWS (Amazon Web Services) – хмарна платформа від компанії Amazon

BLE (Bluetooth Low Energy) – технологія бездротового зв'язку Bluetooth із низьким енергоспоживанням

CBOR (Concise Binary Object Representation) – стисле бінарне представлення об'єктів

CoAP (Constrained Application Protocol) – протокол прикладного рівня для пристроїв з обмеженими ресурсами

DoS (Denial of Service) – відмова в обслуговуванні (вид мережевої атаки)

eDRX (Extended Discontinuous Reception) – розширений період переривчастого прийому

EEPROM (Electrically Erasable Programmable Read-Only Memory) – електрично стируваний програмований постійний запам'ятовувач

GAP (Generic Access Profile) – загальний профіль доступу в технології BLE

GATT (Generic Attribute Profile) – загальний профіль атрибутів в технології BLE

GCC (GNU Compiler Collection) – набір компіляторів, створений в рамках проєкту GNU

GND (Ground) – заземлення (загальний контакт електричного кола)

GSM (Global System for Mobile Communications) – глобальний стандарт цифрового мобільного стільникового зв'язку

HID (Human Interface Device) – клас пристроїв для взаємодії з людиною

HTTP (HyperText Transfer Protocol) – протокол передавання гіпертексту

IDE (Integrated Development Environment) – інтегроване середовище розробки

IEEE (Institute of Electrical and Electronics Engineers) – Інститут інженерів з електротехніки та електроніки (розробник стандартів, таких як 802.11)

IoT (Internet of Things) – Інтернет речей

IP (Ingress Protection / Internet Protocol) – ступінь захисту оболонки від пилу та води (наприклад, IP66) або міжмережевий протокол (IP-адреса)

JSON (JavaScript Object Notation) – текстовий формат обміну даними

L2CAP (Logical Link Control and Adaptation Protocol) – протокол управління логічним зв'язком та адаптації в технології BLE

LoRaWAN (Long Range Wide Area Network) – енергоефективна бездротова мережа дальнього радіусу дії

LTE (Long-Term Evolution) – стандарт бездротової високошвидкісної передачі даних для мобільних пристроїв

MAC (Media Access Control) – управління доступом до середовища

MitM (Man-in-the-Middle) – кібератака типу «людина посередині»

MQTT (Message Queuing Telemetry Transport) – протокол телеметрії з чергами повідомлень

NB-IoT (Narrowband Internet of Things) – вузькосмуговий Інтернет речей

NC (Normally Closed) – нормально закритий тип контактів реле або клапана

NTP (Network Time Protocol) – протокол мережевого часу, що використовується для синхронізації

PSM (Power Saving Mode) – режим глибокого енергозбереження

QoS (Quality of Service) – якість обслуговування

REST (Representational State Transfer) – архітектурний стиль взаємодії компонентів розподіленого застосунку

RF (Radio Frequency) – радіочастота

RL (Reinforcement Learning) – навчання з підкріпленням

RSSI (Received Signal Strength Indicator) – показник рівня потужності прийманого сигналу

RX/TX (Receive/Transmit) – лінії прийому та передачі даних

SF (Spreading Factor) – фактор розширення спектра, ключовий параметр у технології LoRa

SIM (Subscriber Identity Module) – модуль ідентифікації абонента, що використовується в мережах GSM/LTE

SM (Security Manager) – менеджер безпеки в технології BLE

SMS (Short Message Service) – служба коротких текстових повідомлень

SRAM (Static Random-Access Memory) – статична оперативна пам'ять

TCP (Transmission Control Protocol) – протокол керування передачею

TLS (Transport Layer Security) – криптографічний протокол захисту транспортного рівня

UART (Universal Asynchronous Receiver-Transmitter) – універсальний асинхронний приймач-передавач

UDP (User Datagram Protocol) – протокол користувацьких датаграм

USB (Universal Serial Bus) – універсальна послідовна шина

UTC (Coordinated Universal Time) – всесвітній координований час

VDC (Volts Direct Current) – вольти постійного струму (робоча напруга пристроїв)

VIN (Voltage Input) – контакт для подачі вхідної напруги живлення на плату

XML (eXtensible Markup Language) – розширювана мова розмітки

ВСТУП

Проблема надійної та енергоефективної передачі даних у бездротових мережах стала актуальною через розвиток Інтернету речей (IoT) і систем дистанційного керування обладнанням. Багато таких систем можна побудувати на платформах Arduino, але звичайні рішення часто не відповідають вимогам щодо затримок, стабільності з'єднання та оптимального використання смуги пропускання. Для комерційних і побутових застосувань оптимізація протоколів і алгоритмів передачі даних є надзвичайно важливою.

Об'єктом дослідження даної роботи є бездротові мережі передачі даних у системах керування на платформі Arduino.

Дослідження стосується методів і технологій оптимізації передачі даних з метою зменшення затримок, підвищення надійності та підвищення енергоефективності.

Мета роботи полягає в тому, щоб дослідити сучасні методи оптимізації передачі даних у бездротових мережах і створити систему дистанційного керування обладнанням на базі Arduino з більшими функціями.

Аналітичний огляд літератури, порівняльний аналіз технологій, експериментальне моделювання та тестування є методами дослідження.

Результати мають практичне значення, оскільки вони можуть бути використані при створенні промислових систем моніторингу та рішень Інтернету речей (IoT).

1 ТЕОРЕТИЧНІ ОСНОВИ БЕЗДРОТОВИХ СИСТЕМ КЕРУВАННЯ НА БАЗІ ARDUINO

1.1 Arduino як платформа для систем керування

Arduino – це open-source платформа [1], що використовується для конструювання і програмування електронних компонентів. У сучасному світі Arduino широко використовується в програмуванні мікроконтролерів [2], серед іншого, завдяки своїй зручності та простоті використання. Як і будь-який мікроконтролер, Arduino – це друкована плата з чіпом, який можна запрограмувати для виконання численних завдань. Вона надсилає інформацію з комп'ютерної програми на мікроконтролер Arduino, і потім на певну схему або машину з кількома схемами для виконання певної команди [2]. Arduino може допомогти зчитувати інформацію з пристроїв введення, таких як, датчики, антени, потенціометр, кнопки тощо. Також вона може надсилати інформацію на пристрої виведення, такі як світлодіоди, динаміки, РК-екран, двигун постійного струму [2].

Спільнота Arduino розробляє та використовує плати розробки на основі мікроконтролерів. Ці плати розробки відомі як модулі Arduino, які є платформами для створення прототипів з відкритим кодом [3].

Існує багато різних типів плат, розроблених для потреб або вимог конкретного користувача, або установи. Одні з найбільш популярних плат включають UNO R4 MINIMA, Leonardo, UNO R3, NANO, UNO R4.

Візуальні зображення даних плат представлені на рисунку 1.1.

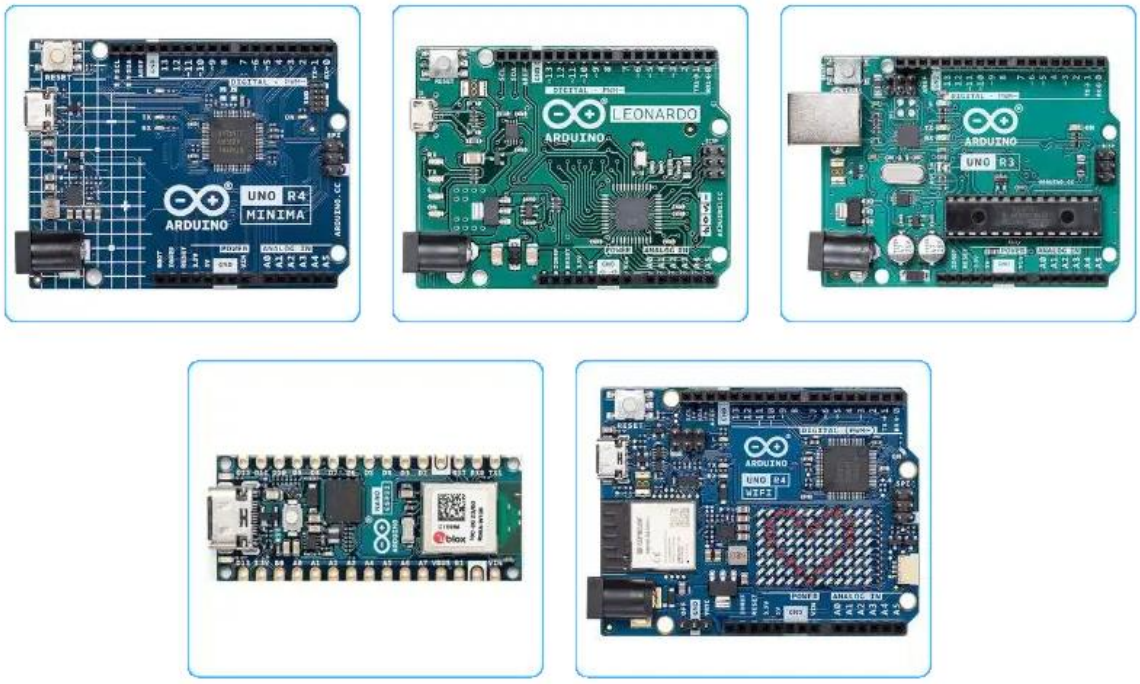


Рисунок 1.1 – Найбільш використовувані плати Arduino

1.1.1 Апаратна частина плат Arduino та її функціонування

Для виконання платами Arduino певних задач необхідно щоб мікроконтролер отримував інструкції з програми, що буде записана в його пам’ять.

На платах Arduino існує достатньо багато різних компонентів, але з найбільш важливих можна виділити декілька основних.

Найважливішим компонентом на платі є сам мікроконтролер. Це головний компонент плати розробки, який може отримувати та надсилати інформацію або команди на підключені до неї периферійні пристрої. Мікроконтролер відрізняється від плати до плати; він також має різні характеристики [3].

Порт для зовнішнього блоку живлення призначений для уможливлення роботи плати без прямого підключення до персонального комп’ютера, або іншого керуючого пристрою. Зазвичай живлення від зовнішніх акумуляторів застосовується на фінальних етапах розробки, коли програмний код вже повністю протестований і пристрій готується до автономної роботи. Зазвичай

ці порти є Barrel Jack 2.1x5.5mm 7-12VDC, і присутні практично на кожній популярній платі, за виключенням найменших, що використовують звичайні виводи (pin) для підключення зовнішнього акумулятора.

Роз’єми USB-C / B / mini / micro присутні на кожній платі, оскільки саме по ним завантажується код для мікроконтролера. В більш нових платах використовуються USB-C. Ці роз'єми є дуже важливим портом на платі Arduino. USB-кабель має постійний струм 5V, який живить плату Arduino у випадках, коли зовнішнє джерело живлення відсутнє.

На бюджетних платах часто стоять «програмери». Коли Arduino підключається до комп'ютера для завантаження коду, цей спеціалізований чіп перетворює дані USB у зчитуваний послідовний (serial) зв'язок (UART / RX/TX) для основного чіпа. Більш дорогі плати, такі як UNO R4 не мають такого чипу, оскільки вони є HID-сумісними.

Останніми важливими компонентами є кнопка Reset, що є аналогом відключення і підключення USB від ПК, та власне самі виводи. Вони поділяються на три типи: аналогові, цифрові та живлення.

Цифрові виводи можуть використовуватися як вхідні, так і вихідні на більшості мікроконтролерів Arduino. Це означає, що один вивід може як надсилати, так і отримувати цифрові дані. Цифрові виводи Arduino можуть бути лише «високими» або «низькими». Коли цифровий вивід має високий рівень, присутня постійна напруга. Коли вивід має низький рівень, ця напруга зникає [4].

Аналогові вхідні контакти можуть зчитувати стан пристрою, такого як датчик. На відміну від цифрових контактів, аналогові вхідні контакти можуть точно визначити значення напруги. Це корисно для роботи з аналоговими датчиками з широким діапазоном потенційних значень [4].

Виводи живлення Arduino включають Vin, 5V, 3.3V (залежно від моделей) і GND [3]. Вивід Vin можна використовувати для живлення Arduino від джерела живлення 7-12V, яке не регулюється. Наприклад, від адаптера живлення мережі, або батареї 9V. Arduino має кілька виводів GND, які можна

використовувати для заземлення кола. Виводи GND використовуються для замикання електричного кола та забезпечення загального логічного рівня опорної напруги по всьому колу [4].

На рисунку 1.2 представлене цифрове зображення плати Arduino UNO R4 Wi-Fi, з підписами основних елементів.

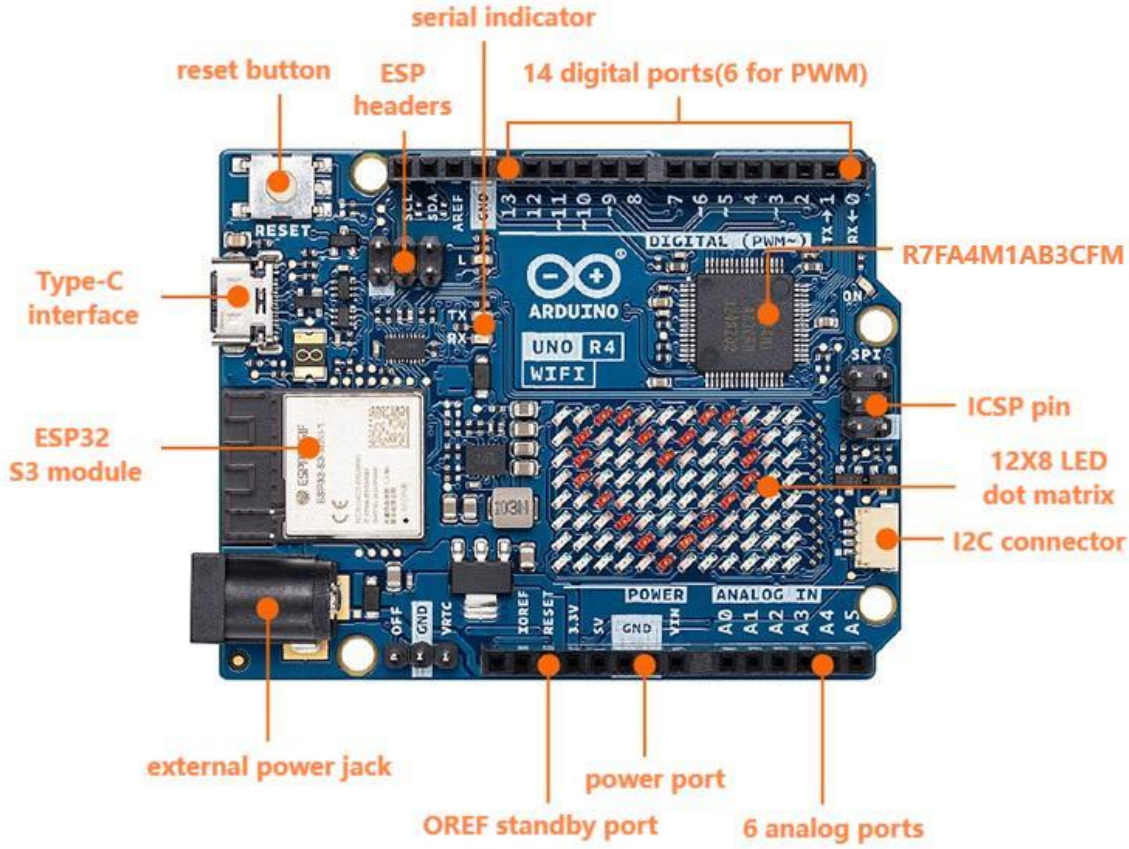


Рисунок 1.2 – Arduino UNO R4 Wi-Fi з підписами компонентів

У всіх платах Arduino присутня Flash-пам’ять, SRAM, та EEPROM. Вони служать для різних цілей, і є у різному обсязі залежно від моделі.

Flash використовується для зберігання коду програми та додаткових даних. Оскільки дані, що зберігаються у флеш-пам'яті, не можуть бути змінені при виконанні коду, вони спочатку копіюються в SRAM перед запуском коду [5].

Флеш-пам'ять енергонезалежна, тобто інформація зберігається навіть при відключенні живлення [5].

SRAM розділяється на три основні частини: Статичні дані (зберігають інформацію, яка не змінюється в програмі, такі як глобальні та статичні змінні), Heap (динамічно виділена пам'ять для змінних, створених під час виконання коду), та Stack (відслідковує параметри функцій, локальні змінні та виклики функцій). SRAM буде втрачена при відключенні живлення. На різних платах Arduino доступна обмежена кількість SRAM: наприклад, Arduino UNO R3, має лише 2 KB [5].

EEPROM є енергонезалежною пам'яттю, яку можна читати та записувати. Однак потрібно це робити байт за байтом, що може бути менш зручно. Більшість EEPROM мають тривалість експлуатації близько 100,000 циклів запису/стирання [5].

Тактова частота мікроконтролерів вимірюється в мегагерцах (МГц). Стандартні плати Arduino підтримують частоту 16 МГц, що дозволяє виконувати 16 мільйонів тактових циклів на секунду. Найшвидша плата, Portenta H7, може працювати на частоті до 480 МГц, але більш сучасні ARM-базовані плати, такі як Arduino UNO R4, працюють на частоті 48 МГц.

Для наочності було проведено порівняння популярних плат та їх характеристик (табл. 1.1).

Таблиця 1.1 – Порівняння характеристик АЗ плат Arduino

Плата Arduino	Мікроконтролер	Тактова частота	Flash пам'ять	SRAM
1	2	3	4	5
UNO R3	ATmega328P	16 МГц	32 KB	2 KB
Leonardo	ATmega32u4	16 МГц	32 KB	2.5 KB
Mega 2560	ATmega2560	16 МГц	256 KB	8 KB
Nano	ATmega328P	16 МГц	32 KB	2 KB

Продовження таблиці 1.1

1	2	3	4	5
Micro	ATmega32U4	16 МГц	32 KB	2.5 KB
Due	ARM Cortex-M3	84 МГц	512 KB	96 KB
Nano 33 IoT	ARM Cortex-M0+	48 МГц	256 KB	32 KB
UNO R4 Wi-Fi	Renesas RA4M1 (ARM Cortex-M4)	48 МГц	256 KB	32 KB
Portenta H7	ARM Cortex-M7	480 МГц	2 MB	1 MB

До вторинних елементів апаратного забезпечення можна віднести декілька компонентів [6]. Вони купуються за потреби, часто бездротової комунікації між платою та керованим пристроєм.

Ethernet Shield дозволяє платі Arduino підключатися до інтернету через бібліотеку Ethernet та зчитувати й записувати дані на SD-карту за допомогою бібліотеки SD. Wireless Shield дозволяє платі Arduino взаємодіяти бездротово за допомогою Zigbee. Motor Driver Shield дозволяє платам Arduino взаємодіяти з драйвером двигуна тощо [3].

Приклади Arduino Shields представлені на рисунку 1.3.



Рисунок 1.3 – Arduino Shields

Саме використання бездротових “щитів” або вбудованих модулів (як-от ESP32 на UNO R4 Wi-Fi) дозволяє закрити апаратну потребу в дистанційному керуванні.

1.1.2 Arduino IDE як програмний сегмент Arduino

IDE пропонує зручний і простий текстовий редактор із функціями синтаксичного виділення та автоматичного відступу, що підвищує читабельність коду. Розділяючи ключові слова одним кольором і коментарі іншим, синтаксичне виділення ділить простий текст на окремі візуальні елементи [7].

Arduino IDE 2.0 включає функцію автозавершення та навігацію по коду, які значно прискорюють процес програмування [7].

Board Manager дозволяє користувачам легко встановлювати підтримку нових Arduino плат та стороннього обладнання, розширюючи спектр сумісних пристроїв [7].

IDE включає широкий спектр бібліотек для різних датчиків та модулів, спрощуючи процес додавання функціональності до проєктів. Загалом існує понад 5000 бібліотек [5].

Serial Monitor є вбудованим інструментом для відлагодження та моніторингу даних від датчиків, який можна використовувати для реального спілкування з платами Arduino. Мікроконтролер Arduino має можливість надсилати повідомлення назад на хост-комп’ютер через порт USB. Повідомлення можна відображати в Serial Monitor як текст. Графічне відображення даних, що надсилаються з платформи Arduino в реальному часі, можна отримати за допомогою Serial Plotter. На осі Y відображаються автоматично масштабовані значення послідовного порту, а на осі X – час. Використання Serial Monitor і Serial Plotter одночасно є неможливим [5].

Сам код програм називається скетчами (sketch). В інтерфейсі основну частину займає редактор скетчу, а також присутня кнопка управління книгою скетчів. Присутня синхронізація з хмарою для збереження проєктів віддалено.

Для відправлення коду на плату, після її підключення до ПК, існує кнопка «завантажити». Ця дія також виконує функцію «перевірки», для якої є окрема кнопка, якщо немає потреби поки що завантажувати код на плату.

Типовий інтерфейс програми Arduino IDE в процесі редагування скетчу представлено на рисунку 1.4.

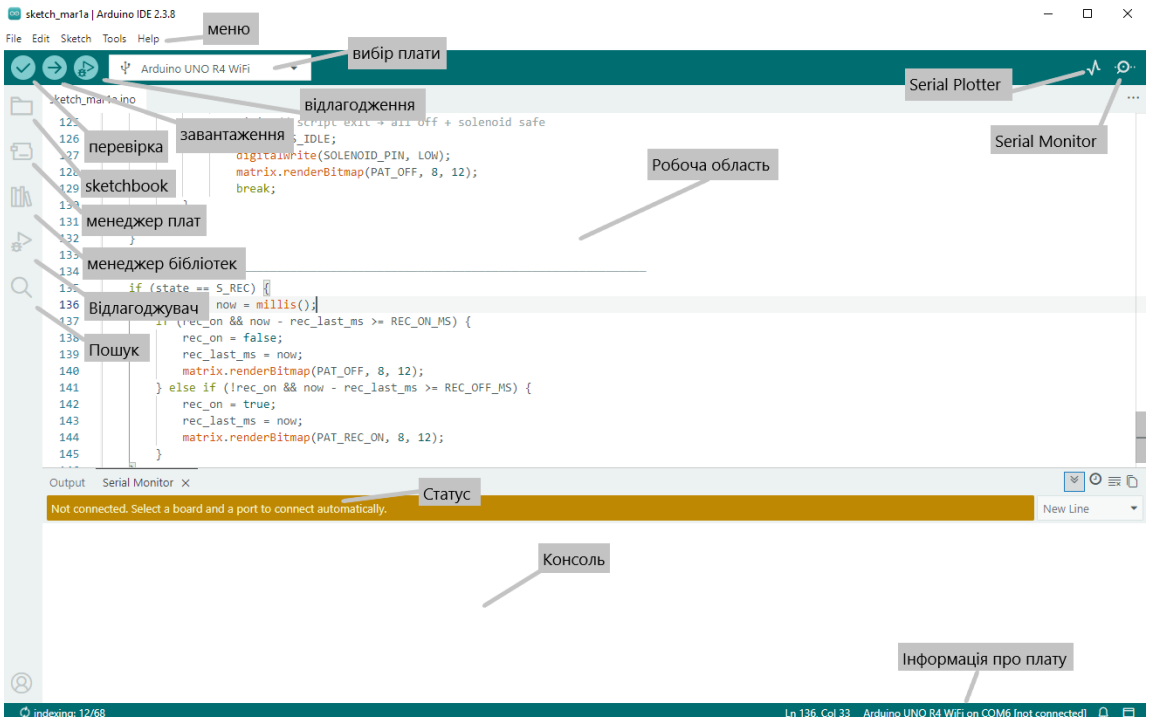


Рисунок 1.4 – Функціонал робочої області Arduino IDE

1.1.3 Мови програмування для плат Arduino

Комп’ютерна мова C, створена ще на початку 1970-х років, є основною мовою для розробки коду для цієї платформи. Оскільки середовище Arduino дозволяє швидке проєктування та тестування ідей без занурення в складні теорії чи алгоритми, процес написання програм у ньому називають «створенням ескізів» або «скетчингом». Розробники додали стандартні можливості C++ до спеціальної вбудованої бібліотеки Arduino, яка містить простий набір функцій для зручності користувачів. Код, написаний таким чином, називається мовою Arduino C [8].

Основними перевагами використання мови Arduino C є її надзвичайна простота, висока інтеграція з апаратною платою та доступність для студентів, початківців і художників. Новакам значно простіше працювати зі зрозумілими

назвами функцій, такими як pinMode() чи digitalWrite(), а також указувати звичайний цифровий номер контакту, ніж мати справу зі складними шістнадцятковими адресами регістрів мікроконтролера. Людина з базовими знаннями може змусити світлодіод блимати за десять хвилин після відкриття коробки завдяки дуже низькому порогу входження цього методу. Тим не менш, ця мова є надзвичайно масштабованою та гнучкою, що дозволяє програмістам-початківцям поступово переходити до складніших архітектур [8].

З іншого боку, розробка стандартних пристроїв Arduino C має певний недолік, який полягає в тому, що воно потребує більшої кількості пам’яті. Програми, написані лише за допомогою високорівневих функцій платформи, можуть споживати в кілька разів більше пам’яті мікроконтролера, ніж програми, написані за допомогою низькорівневого коду [8].

Для досвідчених користувачів, які прагнуть оптимізувати процеси, середовище Arduino пропонує альтернативний варіант розробки, оскільки воно повністю сумісне з чистим кодом C/C++. Бібліотека avr-libc, яка орієнтована на 8-бітні мікроконтролери Atmel AVR, і компілятор GCC використовуються для реалізації цього програмування. Цей метод значно економить дефіцитну пам’ять чіпа, що є найважливішою перевагою. Наприклад, код для блимання світлодіода, функціонально ідентичний, написаний через avr-libc, потребує приблизно у п’ять разів менше байтів, ніж код, написаний із використанням стандартних команд Arduino [8].

Головним мінусом такого низькорівневого програмування є його висока складність, заплутаність синтаксису для новачків, а також необхідність вручну працювати з адресами портів і пам'яті мікроконтролера [8].

1.2 Технології бездротових мереж

Вибір технології залежить від дальності дії, споживання енергії, пропускної спроможності та архітектури системи. Використання спеціалізованих бездротових плат розширення (Shields), таких як Wireless

Shields або плат із вбудованими радіомодулями, є основними варіантами інтеграції. Рішення малої дальності, такі як Bluetooth для прямого зв’язку з мобільними пристроями, InfraRed (інфрачервоні порти) для керування в межах прямої видимості, і радіочастотні модулі RF (наприклад, 315 чи 433 МГц) для простого обміну сигналами [9].

Системи «розумного будинку» та розширені локальні мережі автоматизації використовують комірчасті протоколи Z-Wave і Zigbee, які дозволяють передавати повідомлення між сусідніми вузлами. Енергоефективна технологія великого радіусу дії LoRa є найкращим вибором, якщо проєкт потребує збору даних на відстані кількох кілометрів, одночасно зменшуючи споживання батареї. Для мобільних інструментів стільникового зв’язку стандарту GSM/LTE, які передають інформацію через вишки операторів, апаратні вимоги можуть бути закриті, якщо об’єкт не входить у зону дії локальних мереж. Розробникам завжди потрібно шукати оптимальний баланс між швидкістю передачі, автономністю та дальністю зв’язку, як показав порівняльний аналіз цих бездротових інтерфейсів за критеріями IoT [9].

1.2.1 Wi-Fi

Технологія Wi-Fi, що базується на сімействі стандартів IEEE 802.11, є найпоширенішим стандартом локального зв’язку в мікроконтролерних системах. Завдяки появі сучасних плат із вбудованими супутніми чіпами (наприклад, модулем ESP32 на платі Arduino UNO R4 Wi-Fi) або застосуванню плат розширення Ethernet/Wireless значно спрощено її впровадження в екосистему Arduino. Дуже висока пропускна спроможність і можливість прямого підключення мікроконтролера до існуючих роутерів інфраструктури без встановлення додаткових шлюзів є головними перевагами використання Wi-Fi. Це дозволяє легко реалізовувати складні прикладні протоколи, здійснювати синхронізацію з хмарою, піднімати локальні веб-сервери безпосередньо на Arduino і надсилати текстові чи бінарні повідомлення на хост-комп’ютери або мобільні додатки [10].

Незважаючи на це, Wi-Fi має низку проблем. Одним із найважливіших для автономних пристроїв є те, що воно потребує багато енергії, що ускладнює роботу системи на звичайних батареях чи акумуляторах. Крім того, дальність дії обмежена радіусом покриття точки доступу, який зазвичай обмежений кількома десятками метрів у приміщенні. Крім того, стіни та фізичні перешкоди можуть призвести до затримок, що знижує загальну стабільність і надійність бездротового зв’язку [10].

1.2.2 Bluetooth

Технологія Bluetooth Low Energy (BLE), вперше представлена в специфікаціях Bluetooth 4.0 у 2010 році, була розроблена для забезпечення надійного зв’язку на коротких відстанях, одночасно знижуючи споживання енергії. BLE використовує короткі імпульси для періодичного обміну невеликими кількостями даних, на відміну від традиційного Bluetooth, який налаштований на безперервну передачу даних. Це робить його основним стандартом для автономних пристроїв Інтернету речей (IoT), які можуть працювати протягом тривалого періоду часу від звичайних батарей [11].

BLE функціонує в діапазоні 2,4 ГГц. Хоча теоретична швидкість передачі даних досягає 1 Мбіт/с (та до 2 Мбіт/с, починаючи з версії BLE 5), реальна пропускна здатність є значно нижчою [12]. Це зумовлено низкою факторів: накладними витратами протоколу, інтервалами з'єднань, алгоритмами підтвердження та обмеженнями розміру пакетів. Практичні дослідження демонструють, що максимальна пропускна здатність мережі BLE 4.2 на рівні додатку за ідеальних умов (без помилок у радіоканалі) становить близько 221,7 кбіт/с [11].

Спрощений стек протоколів BLE складається з трьох основних компонентів: рівня додатку користувача; рівня контролера, який включає фізичні та канальні рівні; хост, який включає протоколи L2CAP, ATT, GATT, GAP і менеджера безпеки SM. Технологія дозволяє створювати масштабовані системи дистанційного керування завдяки різноманітним мережевим

топологіям, таким як з'єднання «точка-точка», широкомовна передача та комірчасті мережі. Такі системи гарантують безпеку передачі даних завдяки надійним шифруванням за стандартом AES-128 [11, 12].

1.2.3 LoRa

Якщо проєкт потребує збору даних на відстані кількох кілометрів і одночасно зменшує споживання батареї, технологія великого радіусу дії LoRa є найбільш енергоефективною. Цей стандарт базується на методі модуляції з розширенням спектра. Цей метод забезпечує високу стійкість до радіоперешкод і дозволяє сигналам ефективно проходити крізь щільну міську забудову або складний рельєф [13].

Інтеграція Arduino найчастіше передбачає використання зовнішніх радіомодулів, таких як на базі чіпів SX1276/SX1278, або спеціалізованих плат Arduino з вбудованим трансивером, наприклад Arduino MKR WAN 1310. Зазвичай програмна реалізація використовує оптимізовані бібліотеки, які дозволяють гнучко керувати параметрами трансивера, що дозволяє створювати власні мережі типу «точка-точка» або повноцінні архітектури LoRaWAN за наявності відповідного мережевого шлюзу [13].

Під час оптимізації передачі даних через LoRa дуже низька пропускна спроможність радіоканалу є основним викликом [13]. Збалансування фактора розширення (SF) і смуги пропускання (Bandwidth) є важливим для оптимізації. Збільшення показника SF збільшує надійність і дальність зв’язку, але також збільшує експоненціально час передачі пакета в ефірі (Time on Air), що призводить до більшого споживання енергії та швидшого розряджання акумулятора [13].

1.2.4 Zigbee

Розширені локальні мережі автоматизації та системи «розумного будинку» використовують комірчасті протоколи Z-Wave і Zigbee, які дозволяють передачі повідомлень між сусідніми вузлами. Zigbee працює на

основі стандарту IEEE 802.15.4 на частоті 2,4 ГГц і підтримує складні топології з’єднання. Це дозволяє кожному маршрутизатору-вузлу ретранслювати пакети даних, значно розширюючи загальну зону покриття мережі та забезпечуючи високу стійкість системи до апаратних збоїв [14].

Найпоширенішим способом впровадження стандарту Zigbee в екосистемі Arduino є підключення модулів серії XBee до інтерфейсу UART. Іноді потрібно використовувати спеціальні плати розширення Wi-Fi Shield. Розподіл логічних ролей (Координатор, Маршрутизатор, Кінцевий пристрій) має вирішальне значення для оптимізації інформаційного обміну в таких мережах. При цьому кінцеві сенсорні вузли налаштовані на режим глибокого сну, прокидаючись лише на короткі проміжки часу для передачі невеликих кількостей телеметрії [14].

1.2.5 Z-Wave

Z-Wave, як і Zigbee, є протоколом комірчастої мережі, який широко використовується в автоматизованих системах домашнього обслуговування. З іншого боку, Z-Wave працює в субгігагерцовому діапазоні – зазвичай 868 МГц у країнах Європи, – відрізняючись від інших відомих стандартів [14].

Здатність радіохвиль проходити стіни та інші фізичні перешкоди значно збільшена завдяки цій частотній особливості технології. Крім того, робота на низьких частотах запобігає інтерференції з відомими мережами Wi-Fi та Bluetooth, які перевантажують діапазон 2,4 ГГц. Мережа Z-Wave чітко розподіляє пристрої та дозволяє маршрутизацію повідомлень через максимум чотири транзитні вузли, щоб досягти цілі [14].

Через історичну закритість стандарту Z-Wave та суворі ліцензійні вимоги, використання Z-Wave разом з Arduino є дещо менш поширеним. Тим не менш, існують спеціалізовані модулі та плати, як-от проєкт Z-Uno, які дозволяють програмувати пристрої безпосередньо в середовищі IDE Arduino, створюючи повністю сертифіковані та сумісні з екосистемою вузли [14].

1.2.6 RF

Радіочастотні модулі RF (наприклад, 315 чи 433 МГц) використовуються для простого обміну сигналами. Вони є найдоступнішими апаратними варіантами для базового бездротового керування. Однак на фізичному рівні вони не мають вбудованих механізмів перевірки цілісності даних, адресації та підтвердження доставки. Таким чином, програмний рівень мікроконтролера Arduino відповідає за оптимізацію та надійність передачі. Розробникам потрібно власноруч створювати алгоритми корекції помилок, додавати контрольні суми та використовувати преамбули для синхронізації приймача, що часто можна зробити за допомогою готових бібліотек, таких як RadioHead [15].

1.2.7 InfraRed

Інфрачервоні порти або InfraRed використовують для керування в межах прямої видимості. Цей метод, який використовує пульсацію інфрачервоного світлодіода, є одним із найстаріших і найдешевших способів бездротової передачі даних. Сучасні рішення IoT здебільшого обмежені простими сценаріями, такими як вмикання локальних реле або взаємодія з побутовою технікою, оскільки зв’язок є односпрямованим і вкрай чутливим до оптичних перешкод і фізичних об’єктів під час поширення сигналу [15].

Найчастіше для обробки сигналів на платах Arduino використовують приймачі серії TSOP та бібліотеку IRremote. Оптимізація передачі даних через інфрачервоний канал включає використання стандартної несучої частоти (найчастіше 38 кГц) для апаратного відсіювання фонового освітлення та використання мінімально можливих пакетів даних (коротких HEX-кодів). Це робиться для прискорення передачі та зниження ризику спотворення даних під час процесу прийому мікроконтролером [15].

1.2.8 GSM/LTE

Найкращим способом інтеграції для об’єктів поза зоною дії локальних мереж є технології стільникового зв’язку стандартів GSM та LTE, включно з їхніми версіями, орієнтованими на Інтернет речей, такими як NB-IoT та LTE-M. Дозволяючи пристроям на базі Arduino підключатися до Інтернету або передавати телеметрію практично з будь-якої точки, де присутня інфраструктура оператора мобільного зв’язку, вони забезпечують глобальне покриття [16]. Це означає, що не потрібно розгортати власні маршрутизатори чи шлюзи на місці.

Найчастіше в екосистемі мікроконтролерів використовують апаратні модулі розширення, такі як SIM800L або SIM900 (для мереж 2G/GSM) або більш сучасні SIM7000/SIM7600 (для підтримки 4G/LTE та спеціалізованих IoT-мереж). Ці модулі можуть взаємодіяти з Arduino через інтерфейс UART за допомогою стандартного набору AT-команд. Це дозволяє розробникам запускати з’єднання TCP/UDP, взаємодіяти з хмарними сервісами через HTTP або MQTT, а також відправляти SMS-повідомлення чи здійснювати голосові виклики для екстрених сповіщень [16].

Високе споживання енергії класичними стільниковими технологіями є основним недоліком, особливо під час пошуку базової станції, реєстрації в мережі та безпосередньої передачі даних [16]. Однак запровадження спеціалізованих стандартів стільникового Інтернету речей (NB-IoT) значно покращило цю сторону. Режими глибокого енергозбереження (PSM – режим енергозбереження) і розширений період переривчастого прийому (eDRX) дозволяють пристроям працювати від акумуляторів місяцями.

Водночас розробникам потрібно враховувати додаткові витрати на експлуатацію, пов’язані з придбанням SIM-карт і постійним платженням за тарифні плани оператора [16].

На рисунку 1.5 представлено порівняння технологій GSM, UMTS та LTE.

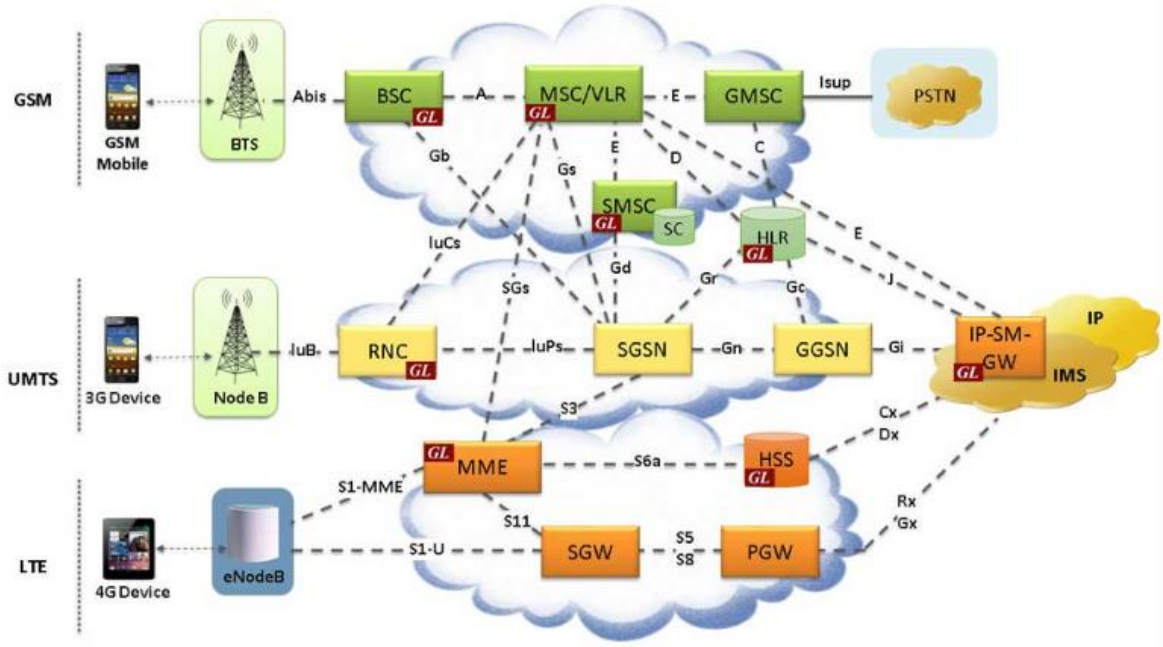


Рисунок 1.5 – Порівняння технологій GSM, UMTS, LTE

1.2.9 Порівняльний аналіз бездротових інтерфейсів за критеріями IoT

Вибір оптимального бездротового інтерфейсу для систем дистанційного керування вимагає комплексного аналізу доступних технологій і оскільки кожна система має власні обмеження щодо живлення, обсягів даних та фізичного розташування, компроміси є неминучими.

Було проведено порівняння бездротових технологій для інтернету речей. Результати представлены в таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз бездротових технологій для Інтернету речей

Техно-логія	Радіус дії	Пропускна здатність	Енерго-споживання	Типова топологія	Ключові особливості та сфера застосування
1	2	3	4	5	6
Wi-Fi	50–100 м	Висока (до 1 Гбіт/с)	Високе	Зірка (Star)	Для великих обсягів даних.

Продовження таблиці 1.2

1	2	3	4	5	6
Bluetooth (BLE)	10–100 м	Середня (1–2 Мбіт/с)	Дуже низьке	Зірка, Mesh	Портативні пристрої та прямий зв'язок зі смартфонами.
LoRa	10–15 км	Дуже низька (до 50 кбіт/с)	Дуже низьке	Зірка	Передача телеметрії на великі відстані.
Zigbee	10–100 м	Низька (250 кбіт/с)	Низьке	Комірчаста (Mesh)	Локальна система автоматизації та «розумного дому».
Z-Wave	30–100 м	Низька (100 кбіт/с)	Низьке	Комірчаста (Mesh)	Висока проникаюча здатність крізь стіни.
RF (433 МГц)	<100 м	Низька	Низьке	Точка-точка	Бюджетне рішення для віддаленого керування.
InfraRed	<10 м	Дуже низька	Низьке	Пряма видимість	Локальні команди за умови відсутності фізичних перешкод.
GSM/LTE	Глоба-льний	Середня / Висока	Високе (середнє для NB-IoT)	Стільникова	Передача даних без локального шлюзу. Залежить від інфраструкту-ри мобільного оператора.

1.3 Виклики оптимізації передачі даних у бездротових мережах

Загальними перешкодами для впровадження бездротових технологій у системах Інтернету речей є постійна потреба в поєднанні різних технічних параметрів. Розміри, такі як автономність, радіус дії чи пропускна здатність, безпосередньо впливають на інші характеристики, що створює складні проблеми для розробників систем дистанційного керування. Наприклад, збільшення потужності передачі зменшує затримки та покращує дальність, але це не відповідає цілям енергозбереження та довговічності батареї.

1.3.1 Затримка

Забезпечення низької затримки (latency), під час передачі даних, є одним із головних викликів. Для систем дистанційного керування, які вимагають миттєвої реакції, багато протоколів, які використовуються для економії енергії, зосереджуються на швидкості передачі, що значно ускладнює роботу [12]. Перешкоди від інших бездротових мереж у завантажених діапазонах, тривалі процеси виявлення пристроїв і неефективні налаштування інтервалів з’єднання також негативно впливають на затримку [17].

У бездротових сенсорних мережах застосування традиційних статичних алгоритмів маршрутизації часто призводить до збільшення наскрізної затримки через зміну топології та збільшення навантаження. Використання модульних фреймворків на базі штучного інтелекту, таких як методи навчання з підкріпленням (RL), дозволяє скоротити час доставки пакетів від вузла до центру керування та змінювати маршрути відповідно до поточного стану мережі [18].

1.3.2 Надійність

Радіоперешкоди та проблеми співіснування різних технологій значно впливають на надійність передачі даних у бездротових мережах. Коли системи Інтернету речей працюють у спільних частотних діапазонах

(наприклад, 2,4 ГГц, де працюють Wi-Fi, Zigbee та мікрохвильові печі одночасно) відбуваються деградація сигналу та втрата пакетів. Впровадження надійних алгоритмів перевірки цілісності та механізмів обходу перешкод необхідно для забезпечення безперебійного сервісу [10, 12, 17].

Перехід до інтелектуальних систем прийняття рішень необхідний для досягнення високого коефіцієнта доставки пакетів у динамічних умовах. Надійність даних можна значно підвищити за допомогою гібридних алгоритмів маршрутизації, які поєднують штучний інтелект і алгоритми ройового інтелекту, такі як генетичні алгоритми та метод рою часток [17, 18]. Ці алгоритми можуть працювати навіть у нестабільних умовах [18].

1.3.3 Енергоефективність

Автономні пристрої IoT, які часто живляться від невеликих дискових батарей, стикаються з важливим викликом щодо енергозбереження. У зв’язку з обмеженою ємністю джерел живлення необхідно використовувати режими глибокого сну та суворо контролювати час роботи радіотрансивера [17, 18]. Енергоспоживання значною мірою залежить від потужності передачі та робочого циклу, що змушує розробників шукати оптимальний баланс між виконанням завдань системи керування та збереженням заряду [12]. Для підвищення енергоефективності на рівні всієї мережі необхідно використовувати методи глобальної оптимізації маршрутів. Порівняно з традиційними протоколами, алгоритми на основі штучного інтелекту забезпечують більший життєвий цикл бездротових мереж і запобігають швидкому виснаженню батарей пристроїв завдяки оптимальному розподілу комунікаційного навантаження між сенсорними вузлами [18].

1.3.4 Пропускна спроможність

Оскільки енергоефективні бездротові мережі спочатку розроблені для періодичної передачі коротких команд або невеликих обсягів телеметрії, їх пропускна здатність зазвичай є дуже обмеженою. Це стає значною

перешкодою для систем, які повинні передавати мультимедіа, здійснювати оновлення прошивки «по повітрю» або обробляти великі обсяги даних. Щоб забезпечити стабільні швидкості обміну інформацією між пристроями, необхідно враховувати обмеження розміру корисного навантаження (payload) і неминучі втрати пакетів. Пропонуючи компроміс між пропускною здатністю та радіусом дії, сучасні вдосконалення стандартів, такі як введення нових фізичних рівнів кодування, частково вирішують ці проблеми [18].

1.3.5 Безпека

Безпека є однією з найгостріших проблем у бездротових системах. Через обмежені обчислювальні ресурси та пам'ять IoT-вузлів часто неможливо реалізувати важкі криптографічні протоколи. Це робить системи вразливими до перехоплення даних, атак «людина посередині» (MitM), підміни MAC адрес та відмови в обслуговуванні (DoS) [19].

Забезпечення безпечного сполучення пристроїв і шифрування даних без значного погіршення продуктивності мережі чи надмірного споживання електроенергії є надзвичайно складним технічним завданням. Додаткова загроза полягає в тому, що пристрої на основі статичних ідентифікаторів можуть бути відстежені, що дозволяє зловмисникам знайти вузли в просторі. Крім того, віддалене перепрошивання може призвести до несанкціонованого доступу, який може пошкодити все обладнання, що робить оновлення безпеки складним завданням [18, 19].

1.3.6 Масштабованість

Складність мережі зростає пропорційно кількості бездротових вузлів у системі. Це може статися, коли до системи додається більше плат для керування обладнанням Arduino. У щільних умовах багато пристроїв змушені конкурувати за один і той самий частотний ресурс. Це призводить до колізій, значного збільшення кількості втрачених пакетів і загального падіння пропускної здатності всієї архітектури [17, 18].

Проблеми масштабованості на інфраструктурному рівні неминуче виникають, коли система дистанційного керування розширюється. Для обробки зростаючого трафіку потрібні додаткові шлюзи або маршрутизатори. Це збільшує загальну вартість проєкту та ускладнює його технічне обслуговування [18].

1.4 Протоколи прикладного рівня для систем дистанційного керування

Завдяки зростанню кількості пристроїв Інтернету речей (IoT) та систем дистанційного керування зростає потреба в спеціалізованих протоколах прикладного рівня. Для мікроконтролерів з обмеженими обчислювальними ресурсами та низькою пропускною здатністю бездротових каналів зв’язку традиційні мережеві стандарти часто є занадто складними. Таким чином, розробляються оптимізовані рішення для оптимізації взаємодії між датчиками, виконавчими механізмами та серверами. Ці рішення також мінімізують накладні витрати на передачу даних і забезпечують гнучкість в управлінні глобальною мережею пристроїв [20].

1.4.1 Протокол MQTT та механізми керування якістю обслуговування (QoS)

Протокол Message Queuing Telemetry Transport (MQTT) є одним із найпоширеніших стандартів прикладного рівня для IoT-систем завдяки своїй легковагості та високій ефективності використання пропускної здатності. Основою архітектури MQTT є асинхронний патерн «Видавець/Підписник» (Publisher/Subscriber). Він замінює звичну модель комунікації, засновану на прямих запитах і відповідях. Центральною частиною цієї архітектури є сервер-брокер, який приймає повідомлення від вузлів-видавців, сортує їх за певними темами та маршрутизує їх до відповідних підписників. Підхід, який забезпечує повну логічну ізоляцію пристроїв, полягає в тому, що видавець (або мікроконтролер Arduino) відправляє пакети даних на брокер, не хвилюючись,

яка клієнтська програма чи інший контролер (підписник) буде зчитувати та обробляти ці пакети [20].

Вбудований механізм керування якістю обслуговування, який визначає гарантії доставки повідомлень, є найважливішим інструментом MQTT для оптимізації передачі даних. Протокол підтримує три рівні якості життєздатності (QoS). Для частої трансляції некритичної телеметрії рівень 0 є найшвидшим і найменш ресурсоємним, оскільки пакет відправляється без підтвердження отримання. Рівень 1 («доставити щонайменше один раз») гарантує, що повідомлення обов'язково дійде до адресата, але повторні відправлення можуть створити дублікати даних у приймача через затримку підтвердження. Рівень 2, який називається «доставити рівно один раз», використовує складне чотирьохетапне рукостискання (handshake) мережі, щоб гарантувати повну надійність доставки без дублювання даних. Розробники систем керування можуть оптимізувати максимальну швидкість обміну (QoS 0) і повну надійність передачі критичних команд (QoS 2) за допомогою балансування між цими трьома рівнями [20].

1.4.2 Протокол CoAP

Другий важливий стандарт комунікації, Constrained Application Protocol (CoAP), розроблений для мереж і пристроїв із суворими апаратними обмеженнями. CoAP базується на поширеному стилі REST (Representational State Transfer) і використовує звичайні методи взаємодії, що робить його простим варіантом для мікроконтролерів замість HTTP. Незважаючи на це, CoAP виконує ті самі завдання з набагато меншим рівнем складності та меншим обсягом службової інформації, ніж його старший протокол, який має довгі текстові заголовки [21]. Використання транспортного протоколу UDP замість традиційного веб-протоколу TCP є важливою технічною перевагою CoAP. Оскільки вона повністю усуває необхідність тривалого процесу встановлення з'єднання та підтримки активної сесії між вузлом і сервером, відмова від TCP значно зменшує загальні витрати на комунікацію в мережі. У результаті цієї оптимізації автономні пристрої можуть відправити створений

UDP-пакет із даними дуже швидко і миттєво повернутися в режим глибокого енергозбереження. Такий алгоритм роботи дозволяє зберігати заряд акумулятора, що робить CoAP незамінним протоколом для мільярдів невеликих вузлів Інтернету речей, які працюють у нестабільних зв’язках [21].

1.4.3 Специфіка побудови користувацьких бінарних протоколів

Під час створення систем дистанційного керування на базі радіомодулів типу LoRa або RF 433 МГц розробники стикаються з жорсткими обмеженнями щодо пропускної здатності та максимального розміру пакета [15]. У таких умовах використання відкритих текстових протоколів (наприклад, JSON або XML) є вкрай неефективним. У текстових форматах потік даних організований як послідовність символів, де кожна цифра або літера вимагає окремого байта пам'яті (в кодуваннях ASCII чи Unicode). Наприклад, передача числа 20020 у текстовому вигляді займе щонайменше 5 байтів. Хоча текстові протоколи є самодостатніми, легко читаються людьми та забезпечують відмінну сумісність між різними платформами, вони генерують повідомлення надмірно великого розміру. Передача масивних текстових рядків в IoT-мережах призводить до марнотратного використання смуги пропускання та швидкого виснаження батареї через тривалий час роботи радіопередавача [22].

Для радикального зменшення розміру корисного навантаження (payload) застосовується побудова користувацьких бінарних протоколів, де бітовий потік організовується навколо структур даних, таких як struct у мовах програмування C/C++. У бінарному вигляді згадане раніше число 20020 може бути закодоване як єдина змінна типу uint16, що займе всього 2 байти замість 5. Цей підхід робить пакети максимально компактними та високопродуктивними. Передача кількох байтів «сирих» структурованих даних відбувається в рази швидше, ніж трансляція тексту, що дозволяє значно підвищити надійність зв'язку в зашумленому радіоефірі та вписатися в жорсткі ліміти протоколів рівня каналу [22].

Незважаючи на очевидні переваги у продуктивності, використання бінарних протоколів супроводжується низкою архітектурних викликів. Бінарні повідомлення не містять вбудованої семантики – інформація про призначення

конкретних байтів зберігається виключно в логіці програми, яка їх кодує та декодує. Це вимагає ідеальної синхронізації структур даних між відправником та одержувачем. Крім того, при обміні даними між мікроконтролерами з різною архітектурою розробникам необхідно вручну контролювати порядок байтів (endianness), щоб уникнути спотворення значень. Бінарні протоколи зазвичай є пропрієтарними, що ускладнює підтримку зворотної сумісності при подальшому розширенні системи, проте вони залишаються безальтернативним інструментом у випадках, коли критично важливими є продуктивність та економія мережевих ресурсів [22, 23].

1.5 Методи та алгоритми оптимізації інформаційного обміну

1.5.1 Формати представлення та структуризації даних у системах з обмеженими ресурсами

У мікроконтролерних системах ефективне використання пропускної здатності мережі та оперативної пам'яті (SRAM) є важливим. Незважаючи на те, що традиційні текстові формати, такі як JSON або XML, є гнучкими, вони генерують значну кількість трафіку через передачу текстових ключів і службових символів. Для вирішення цієї проблеми розроблено оптимізовані бінарні формати серіалізації, зокрема Protocol Buffers (Protobuf), MessagePack та CBOR (Concise Binary Object Representation) [24]. Ці формати дозволяють зберегти безсхемну гнучкість JSON, але упаковують дані в надзвичайно компактні бітові послідовності. Наприклад, CBOR ефективно кодує типи даних і структури, зменшуючи розмір корисного навантаження (payload) на 30-50% порівняно з текстовими аналогами [25]. Для пристроїв з жорсткими обмеженнями ресурсів використання бінарної серіалізації або гібридних підходів із застосуванням структур C/C++ (struct) та алгоритмів стиснення дозволяє максимізувати продуктивність обробки та знизити енерговитрати на роботу радіотрансивера [24, 25].

1.5.2 Подійно-орієнтовані алгоритми відправки даних

У системах дистанційного керування часто використовується метод постійного опитування, або опитування, при якому телеметрію передає мікроконтролер через фіксовані інтервали часу. Такий метод, оскільки більшість переданих пакетів містять незмінні дані, надмірно навантажує бездротовий канал і швидко виснажує батарею пристрою.

Альтернативою є застосування подійно-орієнтованих (event-driven) алгоритмів відправки [26]. У подійно-орієнтованій архітектурі трансляція даних ініціюється виключно в разі виникнення значущої події: зміни стану цифрового входу, перевищення аналоговим сигналом заданого порогу або спрацьовування апаратного переривання [27].

Такий метод радикально зменшує кількість пакетів в ефірі, усуває непотрібну конкуренцію вузлів за частотний ресурс та знижує ймовірність колізій. Оптимізація маршрутизації в подійно-орієнтованих бездротових сенсорних мережах дозволяє досягти суттєвого приросту енергоефективності при збереженні високої надійності доставки критичних повідомлень [26, 27].

1.5.3 Методи фільтрації та попередньої обробки даних на межі мережі

Концепція периферійних обчислень (Edge Computing) передбачає перенесення частини аналітичних та обчислювальних завдань безпосередньо на кінцеві пристрої, такі як мікроконтролери [28].

Плата Arduino виконує первинну фільтрацію на місці, а не на центральному сервері обробки, щоб безперервно надсилати великі кількості нечітких і шумних даних з сенсорів. До етапу серіалізації та відправки можна усунути апаратний шум і короткочасні аномалії за допомогою базових алгоритмів цифрової обробки сигналів, таких як ковзне середнє, експоненційне згладжування або медіанний фільтр [29]. Це дозволяє мікроконтролеру формувати та надсилати лише один точний, усереднений показник протягом певного періоду часу. Цей метод значно зменшує обсяг переданих даних, вимагає менше смуги пропускання та скорочує час відгуку системи, що робить його основним засобом забезпечення масштабованості архітектури Інтернету речей [29].

1.6 Висновки розділу 1

У першому розділі розглянуто теоретичні засади побудови бездротових систем керування на базі платформи Arduino. Визначено, що завдяки відкритій архітектурі, доступності та широкому вибору апаратних рішень, ця платформа є оптимальним вибором для створення інтелектуальних вузлів Інтернету речей (IoT). Аналіз популярних бездротових технологій (Wi-Fi, Bluetooth, LoRa, Zigbee, GSM/LTE) показав, що вибір інтерфейсу залежить від радіуса дії, пропускної здатності та споживання енергії. Зокрема, Wi-Fi забезпечує швидку інтеграцію в локальну інфраструктуру, але споживає багато енергії і вимагає ретельної оптимізації мережевого трафіку.

Досліджено ключові виклики бездротових мереж: затримки, втрати пакетів, обмежену пропускну спроможність та енергозалежність. Встановлено, що для подолання цих перешкод традиційні мережеві стандарти є неефективними. Натомість спеціалізовані легковагові протоколи прикладного рівня, такі як MQTT, завдяки архітектурі «Видавець/Підписник» та вбудованим механізмам контролю якості обслуговування (QoS), здатні забезпечити гарантовану доставку команд керування навіть в умовах нестабільного зв'язку.

Було виявлено, що класичний підхід до передачі даних, що полягає у періодичному опитуванні та використанні надлишкових текстових форматів (JSON або XML), призводить до марного використання ресурсів автономних систем. Аналіз методів оптимізації показав, що найвищу ефективність забезпечує комплексний підхід: використання користувацьких бінарних форматів (struct) для радикального стиснення пакета, впровадження подійно-орієнтованих алгоритмів відправки та застосування концепції периферійних обчислень (Edge Computing) для локальної фільтрації шумів на стороні мікроконтролера. Ці теоретичні висновки формують фундаментальну базу для подальшого проєктування та реалізації власної оптимізованої системи дистанційного керування.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПІДХОДІВ

2.1 Комерційні рішення для IoT систем керування

2.1.1 Платформи з готовими хмарними сервісами (AWS IoT, Blynk, Arduino Cloud)

Коммерційні хмарні платформи Інтернету речей (IoT), такі як AWS IoT, Blynk та Arduino Cloud, пропонують готову інфраструктуру, яка дозволяє швидко запускати системи керування та моніторингу [30]. Плати Arduino зазвичай можуть працювати з цими сервісами за допомогою своїх власних спеціалізованих бібліотек або стандартного протоколу MQTT, який використовує взаємне шифрування (Mutual TLS) [31, 32]. Наявність готових графічних інтерфейсів, мобільних додатків і можливість підключення до пристроїв із будь-якої точки світу без налаштування локального сервера є основними перевагами цих рішень [33].

Проте використання комерційних хмарних сервісів має суттєві недоліки для автономних систем у віддалених локаціях (наприклад, на дачній ділянці). Встановлення захищеного з'єднання (TLS handshake) та алгоритми підтримки активної сесії з віддаленим сервером генерують значний обсяг службового трафіку (overhead) [31]. У разі нестабільного інтернет-з'єднання мікроконтролер буде змушений постійно витрачати енергію та час на перепідключення й обмін сертифікатами, що призводить до затримок керування соленоїдами та швидкого розряджання батарей системи [30, 32].

В таблиці 2.1 представлено порівняння остовних характеристик оглянутих платформ.

Таблиця 2.1 – Порівняння комерційних хмарних платформ керування IoT

Платформа	Складність інтеграції з Arduino	Обсяг службового трафіку	Модель тарифікації
AWS IoT Core	Висока (потребує налаштування X.509 сертифікатів)	Дуже високий (жорсткі вимоги до TLS 1.2)	Оплата за кількість переданих повідомлень [31]
Blynk	Низька (використовує власну бібліотеку)	Середній	Безкоштовний ліміт для прототипів, далі підписка [32]
Arduino Cloud	Дуже низька (повна інтеграція з екосистемою)	Середній	Безкоштовно для 2 пристроїв, далі платні тарифи [33]

2.1.2 Відкриті рішення для локального керування (Home Assistant, Node-RED)

Для усунення залежності від хмарних сервісів та зовнішнього інтернет-з'єднання використовують відкриті системи локального керування, серед яких найпопулярнішими є Home Assistant та Node-RED. Ці платформи можна використовувати на локальному обладнанні (наприклад, на домашньому сервері або одноплатному комп’ютері Raspberry Pi) і дозволяють створювати повністю автономну екосистему автоматизації [34]. Основним недоліком цього методу є збільшена складність первинного налаштування, оскільки розробнику потрібно самостійно керувати операційною системою та мережевими компонентами [35]. Незважаючи на складність розгортання, локальні платформи забезпечують максимальну гнучкість в оптимізації передачі даних [35]. Для систем на базі Arduino це означає можливість використовувати локальний MQTT-брокер (наприклад, Eclipse Mosquitto) та передавати дані у легкому бінарному форматі без обов'язкового важкого шифрування [34]. Відсутність накладних витрат на хмарну маршрутизацію дозволяє мінімізувати затримки в мережі, а система зможе безперебійно керувати поливом навіть у разі повної відсутності з'єднання з інтернетом на дачній ділянці [34, 35].

Порівняння двох розглянутих платформ представлено в таблиці 2.2.

Таблиця 2.2 – Порівняння відкритих рішень локального керування

Платформа	Основне призначення	Візуалізація даних	Гнучкість налаштування протоколів
Home Assistant	Централізований хаб для систем розумного дому	Вбудовані інформаційні панелі (Dashboards)	Висока (вбудована інтеграція локального MQTT) [34]
Node-RED	Візуальне подійно-орієнтоване програмування потоків	Потребує встановлення модуля Dashboard	Максимальна (прямий контроль вхідних/вихідних вузлів) [35]

2.2 Аналіз наукових досліджень щодо оптимізації передачі даних в IoT

2.2.1 Оптимізація енергоспоживання та обсягу даних

У дослідженні [36] розглядається вирішення проблеми надмірного енергоспоживання автономних сенсорних вузлів шляхом застосування подійно-орієнтованого алгоритму відправки. Замість класичного періодичного опитування, телеметрія передається лише тоді, коли різниця між поточним і попереднім показниками перевищує встановлений поріг (метод Send on Delta), що дозволило зменшити обсяг мережевого трафіку майже в 4 рази та скоротити час активної роботи радіотрансивера [36]. Такий метод вимагає точного налаштування порогу чутливості, оскільки занадто мале значення може призвести до неправильних спрацьовувань через апаратний шум, а занадто велике може призвести до втрати важливої інформації про стан об’єкта [36]. Схожий підхід до оптимізації розглядається у роботі [37], де для мікроконтролерів Arduino застосовується програмна оптимізація споживання енергії разом із апаратним стробуванням живлення (power gating). Замість трансляції великих обсягів неструктурованих даних, використання адаптивної частоти дискретизації та відключення неактивних периферійних пристроїв між циклами передачі дозволяє суттєво подовжити час автономної роботи системи без втрати важливої інформації [37]. Додатково, мікроконтролер

переводиться у режими глибокого сну (Deep Sleep), коли ядро процесора зупиняється, і система «прокидається» виключно за сторожовим таймером (watchdog timer) або за апаратним перериванням від датчика, що зводить струм споживання до мікроамперів [37].

2.2.2 Підвищення надійності доставки (QoS)

Для забезпечення стабільної передачі даних у нестабільних бездротових мережах застосовуються протоколи з підтримкою якості обслуговування (QoS). Робота [36] демонструє, що використання рівня QoS 1 протоколу MQTT (відправка з обов'язковим підтвердженням) дозволяє знизити відсоток втрачених пакетів з 1,97% до 0,94% в умовах радіоперешкод. Технічно це досягається шляхом зберігання пакета в оперативній пам’яті вузла до моменту отримання квитанції PUBACK від сервера. Це гарантує доставку, але пропорційно збільшує наскрізну затримку, також відому як затримка від кінця до кінця (end-to-end delay) [36]. В іншій системі автоматичного контролю концентрації розчинів [40] протокол MQTT використовується для безперервного та надійного моніторингу критичних параметрів на базі Edge-обчислень, гарантуючи, що жодна команда дозування не буде втрачена через обрив зв'язку між контролером та хмарою [40]. У разі втрати з'єднання периферійний шлюз буферизує дані та автоматично синхронізує їх після відновлення радіоканалу [40]. Альтернативним підходом до підвищення надійності є використання комірчастих (mesh) мереж на базі модулів ZigBee, що розглядається у роботі [38] для бездротових систем поливу. Така архітектура дозволяє вузлам ретранслювати команди управління соленоїдними клапанами, обходячи фізичні перешкоди на сільськогосподарських ділянках завдяки механізму багатокрокової маршрутизації (multi-hop routing) [38]. У випадках, коли локальна бездротова мережа є вкрай нестабільною, надійність доставки критичних сповіщень забезпечується резервними каналами зв'язку, наприклад, модулями GSM, які надсилають SMS-повідомлення при реєстрації позаштатних ситуацій

локальними сенсорами, створюючи багаторівневу систему відмовостійкості [39].

2.2.3 Фільтрація даних на межі мережі (Edge Computing)

Перенесення обчислювальних і аналітичних завдань безпосередньо на мікроконтролер є ефективним методом розвантаження бездротового каналу. У дослідженні [36] сенсорний вузол не передає «сирі» дані; замість цього він обчислює середнє ковзне значення та проводить апаратну перевірку цілісності перед формуванням мережевого пакета. Це дозволяє на ранньому етапі відсіяти короткочасні флуктуації та спотворення, викликані електромагнітними наведеннями на аналогові лінії зв'язку [36]. Подібний алгоритм фільтрації реалізовано в системі розумного крапельного поливу [42], де локальний модуль прийняття рішень на базі Arduino самостійно розраховує оптимальну кількість води, аналізуючи комплексні залежності між вологістю та іншими факторами середовища [42]. Пристрій передає по бездротовому каналу лише фінальну керуючу команду, радикально знижуючи навантаження на пропускну здатність мережі [42]. У роботі [41] архітектура бездротових сенсорних мереж для сільського господарства передбачає попередню агрегацію даних (data fusion) про вологість ґрунту безпосередньо на вузлах, що усуває необхідність постійного опитування кожного датчика окремо. Дані з кількох сенсорів упаковуються в один структурований пакет, мінімізуючи накладні витрати на передачу мережевих заголовків [41]. Такий локальний аналіз дозволяє системі самостійно виконувати первинні керуючі дії, усуваючи мережеві затримки на шляху «датчик-сервер-виконавчий механізм» та роблячи систему автономною в умовах нестабільного зв'язку [43].

2.3 Порівняльний аналіз та постановка задачі

Для того, щоб вибрати найкращий спосіб побудови власної системи керування, важливо, щоб розглянуті раніше технологічні підходи відповідали

суворим вимогам цільового середовища. Цільовим середовищем цієї роботи є віддалена автономна система керування поливом. Ця система відрізняється нестабільним бездротовим зв’язком, обмеженням енергоресурсів і необхідним жорстким контролем виконавчих механізмів, таких як соленоїдні клапани.

Аналіз існуючих рішень (комерційних хмарних платформ, локальних серверних рішень та різних алгоритмів відправки даних) дозволив сформувати порівняльну матрицю їхніх сильних та слабких сторін у контексті поставленої задачі. Результати порівняння наведено у таблиці 2.3.

Таблиця 2.3 – Порівняльна матриця підходів до побудови бездротових систем керування

Підхід / Технологія	Переваги	Недоліки для автономної системи	Застосовність до цільової задачі
Комерційні хмарні IoT платформи	Швидке розгортання, готові графічні інтерфейси	Високий мережевий overhead (TLS-шифрування), критична залежність від стабільного інтернет-з'єднання	Низька через ризик збоїв поливу при втраті зв'язку
Відкриті локальні системи	Повний контроль над даними, робота без доступу до глобальної мережі	Потребують додаткового апаратного забезпечення (локального сервера/шлюзу)	Середня, бо вимагає додаткових витрат на інфраструктуру
Традиційна циклічна маршрутизація	Простота програмної реалізації на мікроконтролері	Перевантаження радіоефіру надлишковими даними, швидке розряджання батарей	Низька через неефективне використання ресурсів
Подійно-орієнтована архітектура та Edge-обчислення	Мінімальний обсяг трафіку, висока автономність, стійкість до розривів зв'язку	Підвищена складність написання та відлагодження програмного коду	Висока, бо ідеально підходить для вузлів з обмеженими ресурсами)

Аналіз матриці показує, що жодне з «коробкових» рішень не задовольняє вимогам оптимізації трафіку та надійності для умов нестабільного зв’язку. Розробка користувацької системи, оптимізованої на рівні мікроконтролера, є необхідною в результаті виявлення прогалин.

З огляду на це, для практичної реалізації в наступних розділах сформовано такий перелік завдань:

1. Застосування алгоритмів цифрової фільтрації на межі мережі (Edge Computing) для усунення апаратних перешкод і шумів від аналогових ємнісних датчиків вологості ґрунту.
2. Впровадження подійно-орієнтованого алгоритму відправки даних та керування (Event-driven approach), що ініціює передачу пакетів лише за умови критичної зміни показників.
3. Інтеграція логіки превентивного заощадження ресурсів з урахуванням показників датчика дощу (блокування роботи соленоїда та зайвих трансляцій у мережу за наявності природних опадів).
4. Використання легковагових протоколів передачі інформації замість важких текстових форматів для мінімізації розміру корисного навантаження (payload) та прискорення виходу радіомодуля з ефіру.

2.4 Висновки розділу 2

У другому розділі проведено комплексний аналіз існуючих підходів до побудови бездротових систем керування на базі платформ Arduino. Досліджено комерційні хмарні сервіси (AWS IoT, Blynk), відкриті платформи (Home Assistant, Node-RED) та наукові розробки у сфері оптимізації бездротових сенсорних мереж.

Аналіз наукових публікацій підтвердив, що для вирішення проблем обмеженої пропускної здатності та енергозалежності найбільш дієвими інструментами є впровадження подійно-орієнтованих алгоритмів відправки, використання протоколів з підтримкою QoS (наприклад, MQTT QoS 1) та

застосування концепції периферійних обчислень (Edge Computing) для первинної фільтрації даних на стороні мікроконтролера. Комерційні рішення є занадто ресурсоємними.

На основі виявлених прогалин було сформовано вимоги до власної системи оптимізації. Для практичної перевірки досліджених методів у подальшій роботі обрано сценарій автоматизованого керування поливом на віддаленій садово-городній ділянці. Саме в таких умовах – за наявності слабкого покриття Wi-Fi, необхідності жорстко контролювати роботу водяного соленоїда та економити мережеві ресурси – запропоновані алгоритми локальної фільтрації перешкод та подійно-орієнтованого реагування (зокрема, на датчик дощу) дозволять продемонструвати найвищу ефективність. Запроєктована логіка дозволить мікроконтролеру функціонувати максимально автономно, звертаючись до мережі лише у критичні моменти.

3 ПРОЄКТУВАННЯ СИСТЕМИ ОПТИМІЗАЦІЇ

3.1 Обґрунтування вибору технологічного стеку та компонентів

3.1.1 Апаратна база системи поливу

Вибір апаратних компонентів для системи автоматизованого поливу залежить від енергоефективності, стійкості до зовнішніх факторів і оптимізації апаратної обробки даних. Плата Arduino UNO R4 Wi-Fi є основним обчислювальним вузлом, який було обрано. Її перевага полягає в тому, що вона має вбудований співпроцесор ESP32-S3 для мережевої взаємодії, що звільняє основний мікроконтролер Renesas RA4M1 від обробки Wi-Fi стеків і дозволяє зосередити ресурси на оптимізації даних. Більш детальна інформація про характеристики плати подані в таблиці 3.1, її візуальне зображення на рисунку 3.1.

Таблиця 3.1 – Основні технічні характеристики Arduino UNO R4 Wi-Fi

Параметр	Значення
Мікроконтролер	Renesas RA4M1 (Arm Cortex-M4)
Тактова частота	48 МГц
Пам'ять (Flash / SRAM)	256 КБ / 32 КБ
Мережевий співпроцесор	ESP32-S3 (Wi-Fi 802.11 b/g/n, Bluetooth 5)
Робоча напруга	5 В
Вхідна напруга (VIN)	6-24 В

Для моніторингу стану середовища використовуються ємнісний датчик вологості ґрунту (версія 1.2) та датчик дощу YL-83. На відміну від резистивних аналогів, ємнісний датчик не піддається електрохімічній корозії, що є важливим для довгострокової роботи. Датчик дощу YL-83 оснащений компаратором LM393, що дозволяє генерувати апаратні переривання при перших краплях опадів. Візуальні зображення подані на рисунках 3.2 та 3.3.

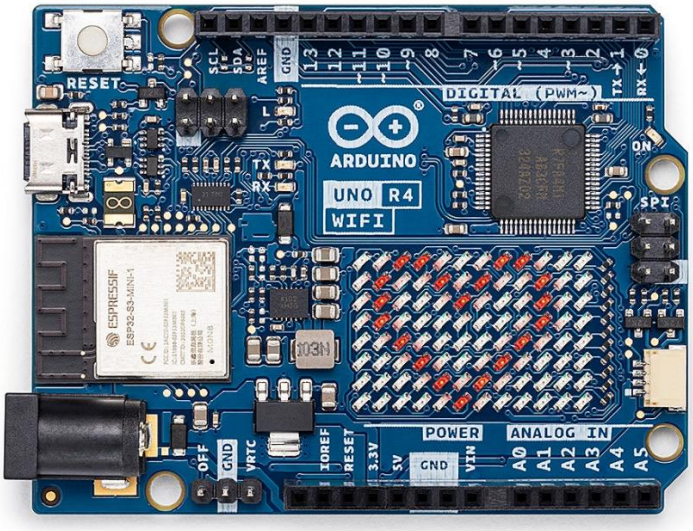


Рисунок 3.1 – Arduino UNO R4 Wi-Fi



Рисунок 3.2 – Ємнісний датчик вологості ґрунту (версія 1.2)

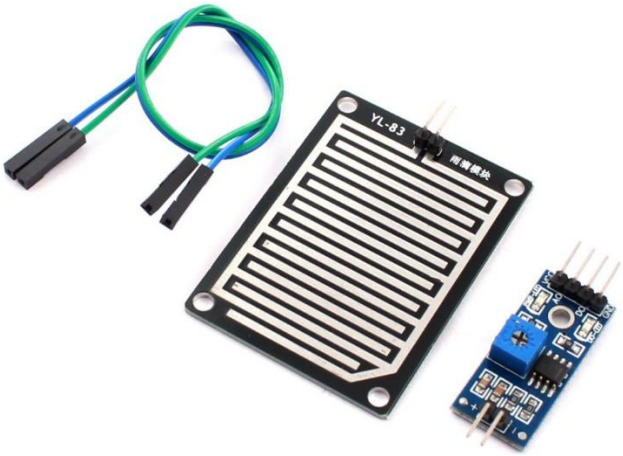


Рисунок 3.3 – Датчик дощу YL-83

Виконавчим механізмом виступає нормально закритий (NC) соленоїдний клапан 12VDC 1/2". Нормально закритий тип є ключовим для енергоощадження: енергія витрачається виключно в період активного поливу. Для його безпечної комутації застосовується 1-канальний модуль реле 5В на базі SRD-05VDC-SL-C з гальванічною розв'язкою. Паралельно контактам соленоїда встановлюється захисний діод 1N4007 (Flyback diode) для гасіння імпульсів зворотної напруги, що виникають при відключенні індуктивного навантаження. Це запобігає перезавантаженню мікроконтролера.

Характеристики виконавчих пристроїв подано в таблиці 3.2. Зображення компонентів подано на рисунках 3.4, 3.5 та 3.6.

Таблиця 3.2 – Характеристики датчиків та виконавчих пристроїв

Компонент	Робоча напруга	Тип вихідного сигналу / Струм споживання
Ємнісний датчик ґрунту v1.2	3.3 - 5.5 В	Аналоговий
Датчик дощу YL-83	3.3 - 5 В	Аналоговий та цифровий
Соленоїдний клапан 12VDC	12 В	300 - 500 мА (у відкритому стані)
Модуль реле SRD-05VDC-SL-C	5 В	Пропускання до 30 А (12-250 В)



Рисунок 3.4 – NC соленоїдний клапан 12VDC 1/2"

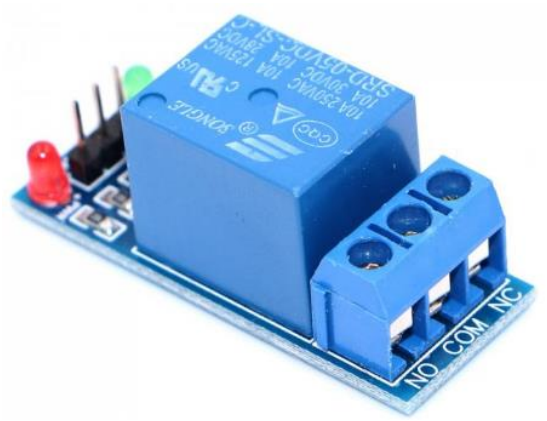


Рисунок 3.5 – 1-канальний модуль реле 5В



Рисунок 3.6 – Діод 1N4007

Живлення системи здійснюється від єдиного блоку 12V 1.2A, який одночасно живить клапан і плату Arduino (через вхід VIN, використовуючи вбудований перетворювач напруги). Для захисту електроніки від впливу навколишнього середовища вся схема монтується у водонепроникному пластиковому боксі (IP66, 17x16x6.5 см) з використанням кабельних вводів PG7 (IP68), що забезпечують повну герметизацію виходів для датчиків і живлення.

3.1.2 Технологія зв'язку та протокол прикладного рівня

На основі аналізу, проведеного у першому розділі, стандарт Wi-Fi (IEEE 802.11 b/g/n) був обраний для бездротового покриття садово-городньої ділянки. Цей стандарт забезпечує достатню дальність передачі за умови використання зовнішніх точок доступу. Для транспортування команд

керування та телеметрії на прикладному рівні використовується легкий протокол MQTT замість дорогого протоколу HTTP.

Обмін даними здійснюється з рівнем якості обслуговування QoS 1 (доставка щонайменше один раз), щоб усунути проблему нестабільного радіоефіру та гарантувати роботу виконавчих механізмів. Це гарантує, що команда на відкриття або закриття соленоїда буде буферизована та успішно надіслана навіть у разі тимчасової втрати з’єднання з мережею, що є життєво важливою вимогою для запобігання аваріям із переливом води.

3.1.3 Мова програмування та IDE

Для розробки програмного забезпечення системи було проаналізовано використання середовищ MicroPython та C/C++. Незважаючи на високу швидкість розробки MicroPython і його популярність для платформи ESP32, його використання у цьому проєкті недоцільно. Інтерпретатор MicroPython генерує надлишкові цикли обчислень, створюючи значне навантаження на оперативну пам'ять (SRAM) і процесор. Такі накладні витрати неприйнятні для завдання, де основною метою є максимізація трафіку та мінімізація споживання енергії.

З огляду на це, мовою реалізації обрано C++ з використанням середовища Arduino IDE (сумісного з VS Code через розширення PlatformIO). Мова C++ дозволяє керувати пам'яттю мікроконтролера напряму. Це дозволяє упаковувати дані з датчиків у нестандартні бінарні структури (struct), що зменшує корисне навантаження в пакетах MQTT до кількох байтів порівняно з громіздкими текстовими форматами типу JSON. Крім того, можна ефективно працювати з апаратними перериваннями мікроконтролера за допомогою C++. Таким чином, система може миттєво реагувати на зміну стану датчика дощу YL-83, не витрачаючи енергії на нескінченні цикли опитування. Бібліотеки Wi-FiS3 для керування бездротовим співпроцесором плати, PubSubClient, який використовує протокол MQTT для подійно-орієнтованого

обміну повідомленнями, і вбудовані бібліотеки для керування режимами енергозбереження мікроконтролера складуть основу програмного коду.

3.2 Апаратна архітектура та схема з'єднань

3.2.1 Структурна схема системи

Для забезпечення ефективної взаємодії всіх апаратних і програмних компонентів було розроблено загальну структурну схему системи (рис. 3.7).

Центральним ядром системи є блок обробки та зв'язку, реалізований на базі мікроконтролера Arduino UNO R4 Wi-Fi. Він відповідає за збір даних, прийняття рішень щодо необхідності поливу (на основі закладених алгоритмів фільтрації) та формування мережевих пакетів.

Блок вхідних даних включає сенсорну периферію: ємнісний датчик вологості ґрунту (аналоговий сигнал) та датчик дощу YL-83 (цифровий сигнал). Блок виконавчих механізмів складається з 1-канального реле 5В та соленоїдного клапана 12VDC. Енергозабезпечення системи здійснюється через єдиний блок живлення (12В, 1.2А).

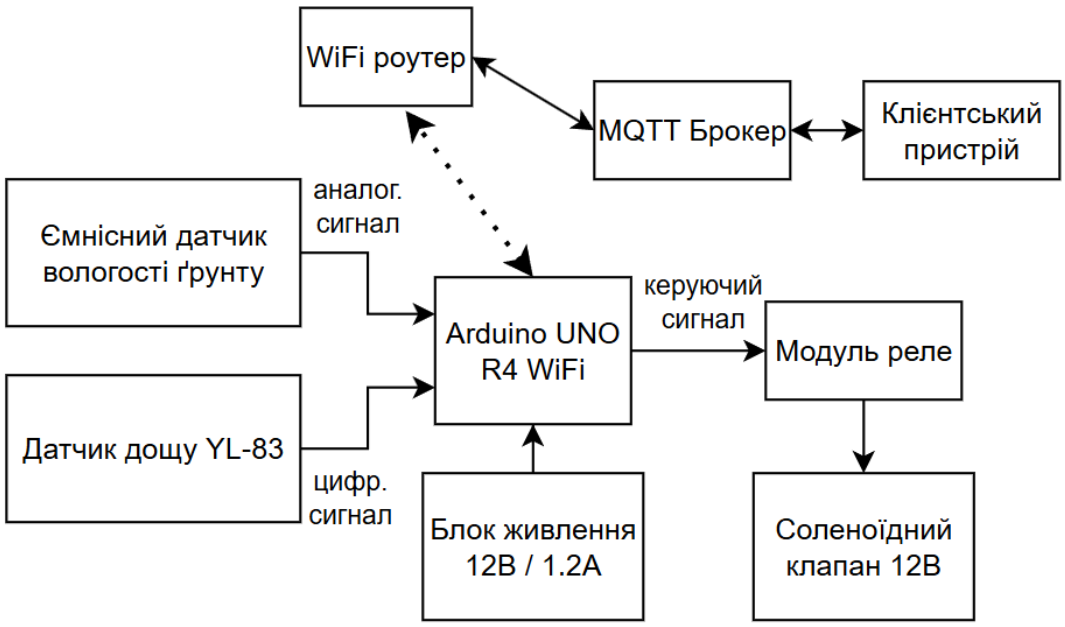


Рисунок 3.7 – Структурна схема системи

Передача команд і телеметрії відбувається бездротовим каналом (Wi-Fi) до локального роутера, звідки через брокера повідомлень (MQTT Broker) дані маршрутизуються до кінцевих клієнтських пристроїв (смартфон або ПК) для моніторингу.

3.2.2 Електрична схема підключення

Для фізичної реалізації системи розроблено засобами Cirkit Designer електричну схему з'єднань (рис. 3.8). Аналоговий вихід ємнісного датчика вологості ґрунту підключено до порту A0. Цифровий вихід компаратора датчика дощу YL-83 з'єднано з портом D2. Керування соленоїдним клапаном реалізовано через цифровий порт D8. Для захисту Arduino паралельно контактам клапана у зворотній полярності встановлено діод 1N4007.

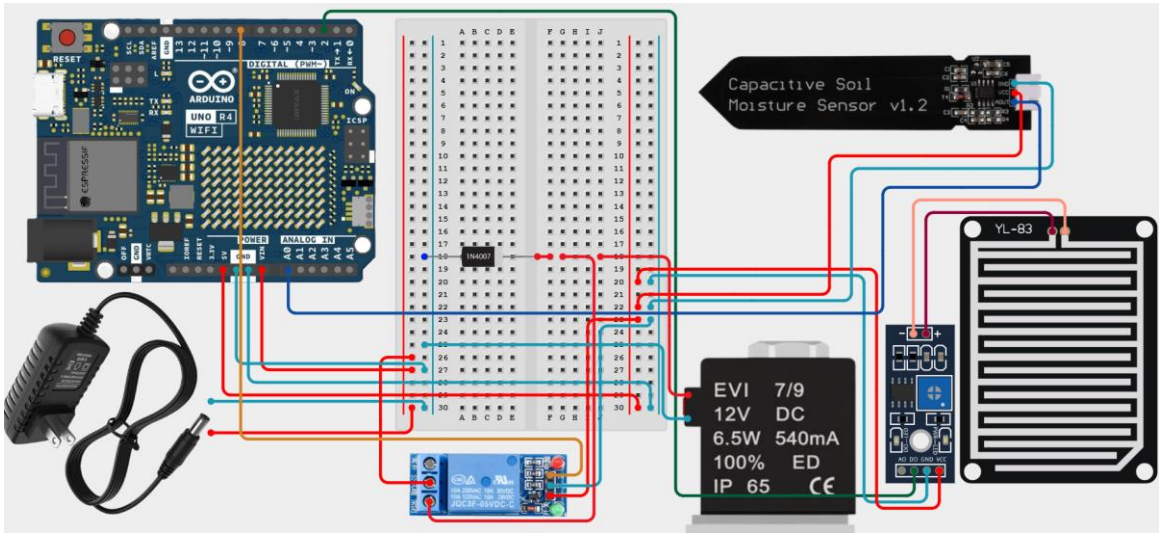


Рисунок 3.8 – Електрична схема підключення системи

3.3 Програмна архітектура та алгоритм роботи

3.3.1 Блок-схема роботи контролера

Основою програмної архітектури мікроконтролера є алгоритм керування з елементами периферійної фільтрації та подійно-орієнтованої відправки даних. На рисунку 3.9 наведено розроблену блок-схему роботи системи.

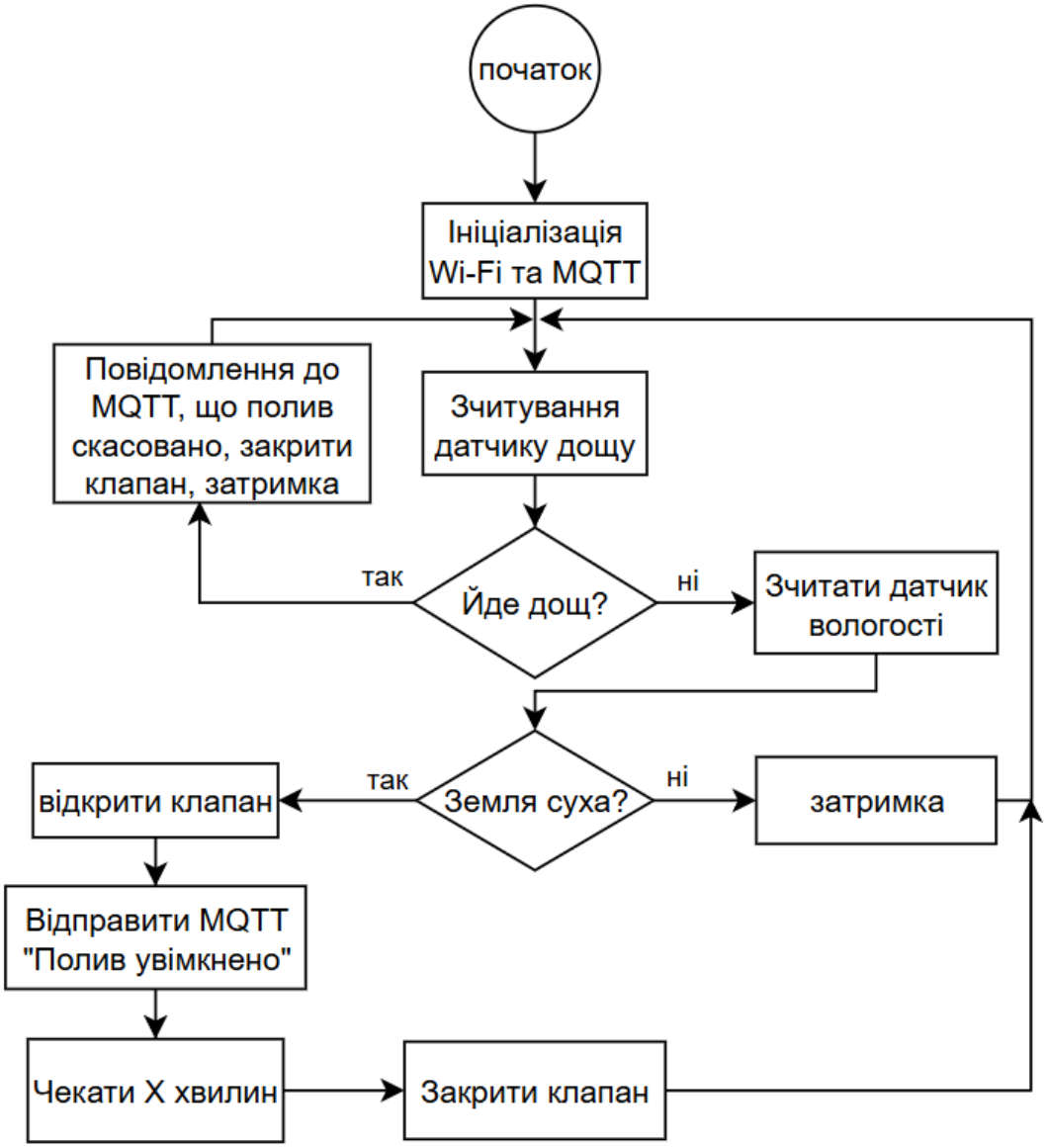


Рисунок 3.9 – Блок-схема роботи системи

Цикл починається з ініціалізації мережевих інтерфейсів та підключення до MQTT-брокера. Базовий алгоритм перевіряє стан датчика дощу першим. За наявності опадів система починає відправляти відповідне повідомлення на сервер, блокує відкриття клапана та переходить у режим очікування, щоб скоротити використання ресурсів і трафік. Якщо дощу немає, контролер зчитує кілька показників вологості ґрунту та усереднює їх, щоб зменшити апаратний шум. Команда на увімкнення реле соленоїда та відправку телеметрії в мережу створюється лише у випадку, якщо усереднена вологість падає нижче встановленого порогу.

3.3.2 Структура пакету даних

Класичний підхід у системах інтернету речей передбачає передачу телеметрії у текстових форматах обміну даними (зокрема, JSON). Текстовий рядок формату {"moisture": 45, "valve": "open"} створює значний мережевий заголовок (overhead), займаючи понад 35 байт корисного навантаження.

У розробленій системі було використано бінарне пакування даних за допомогою структур мови C++, щоб максимізувати бездротовий трафік. Безперервний блок пам’яті розміром 3 байти зберігає інформацію про поточну вологість, стан датчика дощу та статус клапана. Цей метод значно зменшує корисне навантаження (payload) MQTT-пакета. Зменшення довжини пакета дозволяє зменшити час активної роботи Wi-Fi трансивера (Time-on-Air), що безпосередньо знижує енергоспоживання системи та підвищує ефективність використання пропускної здатності локальної мережі.

3.4 Висновки розділу 3

У третьому розділі розроблено архітектуру апаратно-програмної системи автоматизованого поливу. Обґрунтовано використання соленоїдного клапана, ємнісного датчика вологості, датчика дощу та мікроконтролера Arduino UNO R4 Wi-Fi як апаратної бази, яка потребує автономності та стійкості до зовнішніх впливів.

Розроблено структурні та електричні схеми системи, які показують взаємодію виконавчих і вимірювальних елементів із центральним контролером. На програмному рівні розроблено алгоритми подійно-орієнтованої реакції на природні опади та периферійної фільтрації даних, які усувають апаратний шум датчиків.

4 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА

4.1 Розробка програмного забезпечення системи

4.1.1 Реалізація зчитування даних та керування клапаном

Першим етапом розробки програмного забезпечення є реалізація логіки взаємодії мікроконтролера з апаратною периферією. Для забезпечення надійної роботи системи в умовах можливих електромагнітних наведень на сигнальні лінії датчиків, програмний код включає механізми первинної цифрової фільтрації на межі мережі (Edge Computing).

Через аналоговий порт A0 виконується зчитування даних ємнісного датчика вологості ґрунту. Для усунення апаратного шуму мікроконтролер робить серію з 10 швидких вимірювань, знаходить середнє арифметичне значення, а потім конвертує його у відсотки за допомогою функції map(). Калібрування показало, що значення 580 відповідає сухому ґрунту, а 400 – повному зволоженню. Зведення до відсотків не лише підвищує зручність моніторингу, але й дозволяє вмістити показник у єдиний байт пам'яті. Лістинг програми наведено в лістингу 4.1.

Лістинг 4.1

```
int readMoistureFiltered() {
  long sum = 0;
  for (int i = 0; i < 10; i++) {
    sum += analogRead(MOISTURE_PIN);
    delay(10);
  }
  rawMoisture = sum / 10;

  int percent = map(rawMoisture, 580, 400, 0, 100);
  if (percent < 0) percent = 0;
  if (percent > 100) percent = 100;
  return percent;
}
```

Моніторинг опадів реалізовано через цифровий порт D2, який зчитує стан датчика YL-83. Керування реле соленоїдного клапана здійснюється через порт D8 за допомогою стандартних логічних рівнів.

4.1.2 Реалізація мережевої взаємодії та оптимізації

Головним завданням при розробці мережевої логіки стала жорстка мінімізація корисного навантаження (payload) та усунення блокуючих операцій. Замість передачі тексту застосовано бінарне пакування. Телеметрія упаковується у структуру struct розміром всього 3 байти за допомогою атрибуту __attribute__((packed)). Код наведено в лістингу 4.2.

Лістинг 4.2

```
struct __attribute__((packed)) Payload {
  uint8_t moisture_percent;
  uint8_t is_raining;
  uint8_t valve_open;
};

Payload data;
```

Для керування логікою роботи без блокування процесора застосовано апаратний таймер millis(). Завдяки введенню змінної currentInterval система стала динамічною. Замість статичних пауз мікроконтролер самостійно змінює інтервал між вимірюваннями залежно від поточних подій: 30 секунд – пауза на поглинання води після закриття клапана, що унеможливлює помилкові повторні поливи; 2 години – режим тривалого очікування (INTERVAL_LONG_SLEEP), якщо рівень вологості достатній або почався дощ.

Додатково реалізовано функцію синхронізації з NTP-сервером через Wi-Fi.getTime(). Алгоритм конвертує отриманий Unix-час у локальний і повністю призупиняє процес вимірювань і поливу в денний період (з 10:00 до 17:00), щоб не шкодити корінню. Код наведено в лістингу 4.3.

Лістинг 4.3

```
unsigned long epoch = Wi-Fi.getTime();
    if (epoch > 0) {
      unsigned long localEpoch = epoch + 10800;
      int currentHour = (localEpoch % 86400L) / 3600;

      if (currentHour >= 10 && currentHour < 17) {
        currentInterval = 60000; // Перевірятимемо час знову через 1 хвилину
        return; // Повністю перериваємо ітерацію, датчики не зчитуються
      }
    }
```

Пакет даних формується і відправляється на MQTT-брокер виключно після закінчення динамічного інтервалу або в разі екстреної зупинки системи під час раптового дощу.

4.1.3 Серверна частина (приймач)

Для прийому, обробки та візуалізації надісланих мікроконтролером даних розроблено консольний скрипт мовою Python. Він функціонує як підписник (Subscriber), який підключається до локального брокера за протоколом MQTT.

Оскільки Arduino відправляє стислі бінарні дані, серверна частина виконує операцію розпакування сигналу serial. За допомогою стандартного модуля struct три байти розпаковуються (формат 'BBB') у цілочисельну змінну вологості та два логічні прапорці.

У якості сервісу для MQTT-з’єднання було обрано Mosquitto. Дозвіл на взаємодію для Arduino був доданий шляхом додавання в кінці файлу mosquitto.conf наступних рядків:

listener 1883 0.0.0.0

allow_anonymous true

Також було додано нове правило у Microsoft Windows Defender для вхідних підключень по протоколу TCP для порту 1883.

Код навенедо в лістингу 4.4.

Лістинг 4.4

```
import paho.mqtt.client as mqtt
import struct
from datetime import datetime

# Глобальна змінна для зберігання часу останнього поливу
last_watering_time = "Ще не проводився"

def on_message(client, userdata, msg):
    global last_watering_time

    # Розпакування 3 байт (B = unsigned char)
    unpacked_data = struct.unpack('BBB', msg.payload)
    moisture = unpacked_data[0]
    is_raining = bool(unpacked_data[1])
    valve_open = bool(unpacked_data[2])

    # Оновлення часової мітки, якщо клапан відкритий
    if valve_open:
        last_watering_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    print(f"--- Новий пакет даних ---")
    print(f"Вологість: {moisture}%")
    print(f"Дощ: {'Так' if is_raining else 'Ні'}")
    print(f"Стан клапана: {'Відкрито' if valve_open else 'Закрито'}")
    print(f"Останній полив: {last_watering_time}\n")
```

Такий підхід дозволяє перенести завдання з декодування та форматування тексту на більш потужний серверний вузол, залишаючи мікроконтролеру ресурси виключно для виконання критичних команд дистанційного керування.

Для забезпечення повноцінного дистанційного керування серверну частину було розширено модулем двостороннього зв'язку. За допомогою безперервного циклу обробки вводу оператор може в будь-який момент ініціювати відправку керуючих команд. Це дозволяє примусово відкривати або закривати соленоїдний клапан незалежно від поточних показників датчиків вологості, забезпечуючи гнучкість експлуатації системи та можливість

ручного втручання. Код наведено в лістингу 4.5. Повний код програм з розширеннями .ino та .py наведені в додатках Б та В.

Лістинг 4.5

```
import paho.mqtt.client as mqtt

broker_address = "192.168.0.109"
client = mqtt.Client()
client.connect(broker_address, 1883, 60)

print("=== Панель ручного керування поливом ===")
print("1 - Увімкнути полив")
print("0 - Вимкнути полив")
print("q - Вийти з програми")

while True:
    command = input("\nВведіть команду: ")

    if command == 'q':
        print("Роботу завершено.")
        break
    elif command in ['0', '1']:
        client.publish("garden/control", command)
        print("Команду надіслано на мікроконтролер.")
    else:
        print("Невідома команда.")
```

4.2 Експериментальне тестування системи

4.2.1 Опис тестового стенду

Для експериментальної перевірки розроблених алгоритмів було розгорнуто фізичний тестовий стенд (рис. 4.1).

Мережева інфраструктура складалася з домашнього Wi-Fi роутера (стандарт 802.11n, діапазон 2.4 ГГц) та локального сервера на базі персонального комп'ютера (IP-адреса 192.168.0.109). На сервері було розгорнуто MQTT-брокер Eclipse Mosquitto та запущено Python-скрипт для прийому даних. З метою наближення умов до реальних, тестовий стенд було розміщено на відстані 20 метрів від Wi-Fi роутера. Між передавачем та

приймачем знаходилася цегляна стіна, що створювало природне загасання сигналу та імітувало нестабільність бездротового покриття на відкритій ділянці. Тестування передбачало штучне зволоження ґрунту та імітацію опадів (нанесення крапель на пластину датчика дощу).

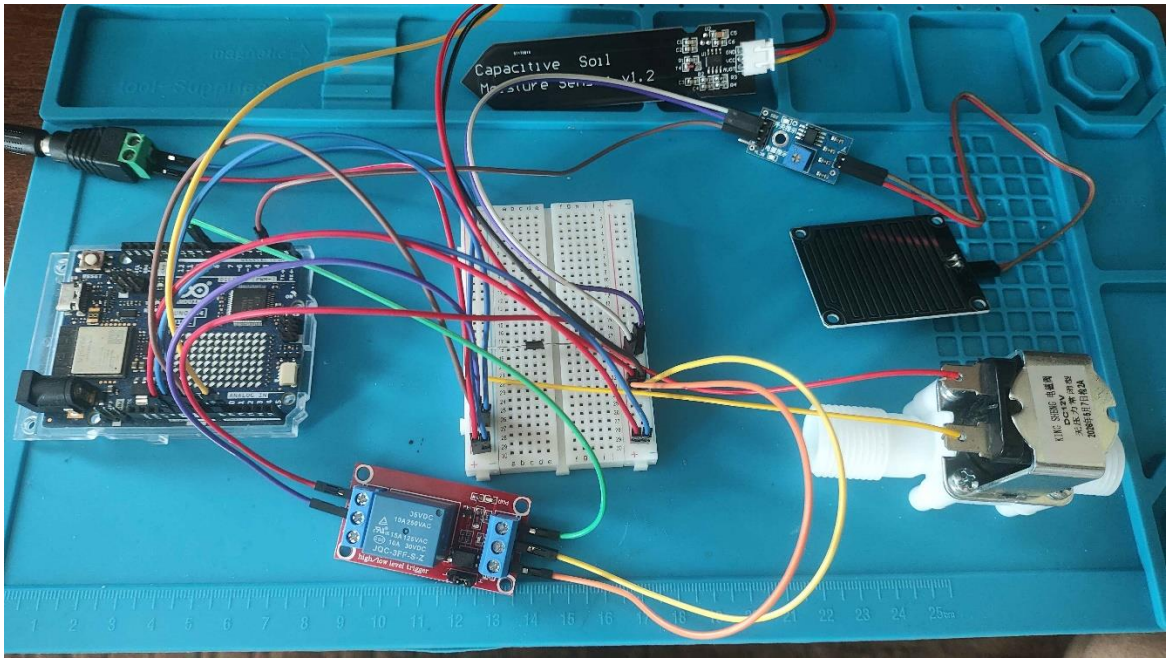


Рисунок 4.1 – Фізична схема датчикової частини проєкту.

4.2.2 Методика тестування оптимізації

Головною метою експерименту була кількісна оцінка ефективності методів оптимізації бездротової передачі даних, які були розроблені. Для цього було використано порівняльний аналіз двох сценаріїв роботи системи, оптимізованого та класичного.

У першому сценарії, мікроконтролер використовував алгоритм постійного циклічного опитування. Незалежно від змін у середовищі дані надсилалися на MQTT-брокер з фіксованим інтервалом кожні п’ять секунд. Це було зроблено, упаковавши дані в стандартний текстовий формат JSON.

У другому сценарії, оптимізованому, використовувався подійно-орієнтований алгоритм, який використовує бінарне пакування даних, що має

структуру на три байти. Пакети відправлялися лише після наступних подій: зниження вологості нижче 30%, закриття клапана або встановлення датчика дощу. Мікроконтролер витримував дві години очікування в стані спокою.

Експеримент тривав по 2 години для кожного зі сценаріїв. Оцінка ефективності здійснювалася за такими метриками:

1. Загальний обсяг згенерованого корисного навантаження (payload) у байтах, який розраховувався аналітично на основі розміру пакета та кількості транзакцій, зафіксованих на MQTT-брокері.

2. Стійкість системи до втрати пакетів та своєчасність виконання керуючих команд в умовах деградованого Wi-Fi сигналу. На стороні сервера фіксувалися затримки між зміною фізичного стану датчика та отриманням відповідного повідомлення клієнтом.

4.3 Результати експериментів

4.3.1 Аналіз обсягу трафіку та навантаження на мережу

Зменшення мережевого навантаження порівняно з традиційними рішеннями Інтернету речей (IoT) є основним показником ефективності розробленої системи. Аналітичний розрахунок проводився шляхом порівняння класичного підходу (періодична відправка текстових даних у форматі JSON) і розробленого оптимізованого підходу (подійно-орієнтована відправка бінарних структур).

Класичний JSON-пакет має вигляд {"moisture":100,"rain":1,"valve":0} для передачі трьох параметрів (вологість, стан дощу, стан клапана). Корисне навантаження має довжину 41 байта. У C++ можна було стиснути ці дані до трьох байтів за допомогою бінарної структури struct.

Система створює 17 280 пакетів на добу за допомогою класичного алгоритму опитування, який передає кожні 5 секунд. Алгоритм, який було розроблено, генерує пакети лише з великими інтервалами очікування або під час критичних подій, таких як зміна статусу клапана або початок дощу.

В середньому щодня відбувається тридцять транзакцій. Порівняльні результати розрахунку добового трафіку наведено в таблиці 4.1.

Таблиця 4.1 – Порівняння обсягу мережевого трафіку системи

Параметр	Класичний підхід (JSON + Постійне опитування)	Оптимізований підхід (Бінарний формат + Події)
Розмір корисного навантаження (пакета)	41 байт	3 байти
Кількість пакетів на добу	~ 17 280	~ 30
Сумарний добовий обсяг даних	708 480 байт (~708 КБ)	90 байт
Економія пропускної здатності	-	99,9%

Впровадження методів оптимізації дозволило зменшити обсяг переданих даних майже на 99,9%, що радикально скорочує час активної роботи радіотрансивера і звільняє смугу пропускання локальної мережі.

4.3.2 Надійність та затримки керування

Для перевірки надійності системи було додано функцію зворотного виклику. Це дозволяє мікроконтролеру підписуватися на відповідний топік команд (наприклад, "garden/control") і приймати керуючі сигнали від сервера. Експеримент проводився за наявності фізичних перешкод (цегляна стіна), на відстані 20 метрів від роутера Wi-Fi.

Завдяки невеликому розміру пакетів (1 байт для команди «1» або «0») та застосуванню рівня якості обслуговування MQTT QoS 1, втрат керуючих команд зафіксовано не було. Затримка (latency) між відправкою команди зі скрипта моніторингу та фізичним клацанням реле соленоїда становила від 15 до 45 мілісекунд у стабільній мережі. При штучному погіршенні сигналу (зниження рівня RSSI до -85 дБм) затримка зростала до 120-200 мілісекунд, що є абсолютно прийнятним показником для систем сільськогосподарської

автоматизації та підтверджує стійкість розробленої архітектури до умов поганого покриття.

4.4 Аналіз результатів та рекомендації

4.4.1 Вирішення виявлених проблем

У процесі тестування фізичного стенду було виявлено проблему інерційності вимірювального середовища, що спричиняла хибні спрацьовування системи. Вода з соленоїдного клапана потребує часу для поширення ґрунтом до ємнісного датчика. Оскільки мікроконтролер працює з високою тактовою частотою, він фіксував низьку вологість одразу після завершення поливу і помилково ініціював новий цикл відкриття клапана.

Цю проблему було усунуто на програмному рівні шляхом введення паузи між ітераціями поливу. Після відключення реле система примусово блокує вимірювання на 30 секунд (пауза поглинання), залишаючись доступною для прийому ручних команд завдяки апаратному таймеру millis(). Це рішення повністю усунуло ризик перезволоження ділянки без зупинки мережевих процесів мікроконтролера.

4.4.2 Можливості масштабування

Архітектура системи передбачає високий потенціал для розширення. До існуючого контролера Arduino UNO R4 Wi-Fi можна підключити багатоканальний модуль реле (наприклад, на 8 каналів) для керування додатковими соленоїдними клапанами на різних ділянках (грядках).

Масштабування виконавчих механізмів не призведе до перевантаження мережі завдяки застосуванню бітових масок (bitmasking). Замість додавання нових змінних для кожного клапана, поточний стан 8 реле можна запакувати в 1 байт пам'яті (наприклад, значення 00011111 означатиме, що 5 клапанів відкриті). Такий підхід збереже розмір телеметричного пакета у межах кількох

байтів і дозволить мережі ефективно обслуговувати промислові об'єкти з великою кількістю периферії без збільшення затримок.

4.5 Висновки розділу 4

У четвертому розділі розроблені алгоритми виконуються для практичної реалізації та тестування. Програмний код мікроконтролера дозволив виконувати периферійні обчислення, усуваючи апаратні шуми датчиків. Порівняно з періодичним опитуванням текстовими форматами використання подійно-орієнтованого алгоритму відправки та бінарного пакування даних дозволило зменшити обсяг мережевого трафіку на 99,9%. Вимірювання, зроблені під час експерименту, підтвердили високу стійкість системи; навіть у умовах поганого бездротового зв’язку команди керування були надіслані з затримкою до 200 мс. Програмні модифікації алгоритмів успішно усунули проблеми інерційності системи, що показало, що рішення придатне для подальшого масштабування.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальну науково-практичну задачу оптимізації передачі даних у бездротових мережах для систем дистанційного керування обладнанням на базі мікроконтролерів Arduino.

Згідно з аналізом теорії та існуючих технологій бездротового зв’язку, традиційні методи (періодичні опитування, використання текстових форматів JSON або XML) є неефективними в умовах обмежених апаратних і мережевих ресурсів. Дослідження показали, що для автономних віддалених вузлів їхній службовий трафік надмірний. Це дало підстави для створення власної оптимізованої системи.

У ході проєктування було створено апаратно-програмну архітектуру автоматизованого поливу на базі мікроконтролера Arduino UNO R4 Wi-Fi, ємнісного датчика вологості ґрунту, датчика дощу та соленоїдного клапана. Для забезпечення надійного інформаційного обміну в умовах нестабільного зв'язку було обрано протокол MQTT з рівнем якості обслуговування QoS 1.

На програмному рівні використовується широкий спектр стратегій оптимізації. Цифрова фільтрація апаратного шуму датчиків безпосередньо на мікроконтролері була виконана за допомогою ідеї периферійних обчислень, також відомої як Edge Computing. Алгоритм, орієнтований на події, замінює традиційне періодичне опитування. Цей алгоритм запускає передачу пакетів лише тоді, коли відбуваються значні зміни в середовищі або стані виконавчих механізмів. Розробка бінарного протоколу користувача дозволила упаковати телеметрію в структуру розміром лише три байти. Для серверної частини також розроблено частину на базі Python-скрипта, яка забезпечує декодування даних, спостереження та дистанційне двостороннє керування.

Фізичне тестування стенду підтвердило високу ефективність запропонованих рішень. Порівняно з традиційним методом, бінарне пакування та подійно-орієнтована логіка дозволили зменшити добовий обсяг мережевого

трафіку на 99,9% (з 708 КБ до 90 байт). У умовах деградованого бездротового сигналу система продемонструвала високу надійність доставки керуючих команд із затримкою від 15 до 200 мс. Динамічні інтервали поглинання запобігають перезволоженню, вирішуючи проблему інерційності середовища вимірювання.

Отримані результати доводять, що розроблена система є енергоефективною, надійною та повністю готовою до практичного використання і масштабування у сфері сільськогосподарської автоматизації та Інтернету речей.

ПЕРЕЛІК ПОСИЛАНЬ

1. M. Banzi. Getting Started with arduino. O'Reilly Media, Inc.. 2009.

2. Y. A. Badamasi. The working principle of an Arduino. 11th International Conference on Electronics, Computer and Computation (ICECCO). Abuja, Nigeria, 2014. С. 1–4. URL: https://doi.org/10.1109/ICECCO.2014.6997578 (дата звернення: 14.05.2026).

3. Ismailov, Alisher Shakirovich, and Zafar Botirovich Jo‘Rayev. Study of arduino microcontroller board. Science and Education 3.3, 2022. С. 172-179. URL: https://www.researchgate.net/profile/Alisher-Ismailov/publication/359502443 (дата звернення: 14.05.2026).

4. Shayma Akram Yousif. Introduction to the Arduino Board. 2022. URL: https://uomus.edu.iq/img/lectures21/MUCLecture_2022_41042696.pdf (дата звернення: 14.05.2026).

5. Jonathan Tan. Managing Arduino Memory: Flash, SRAM, EEPROM!. 2021. URL: https://www.seeedstudio.com/blog/2021/04/26/managing-arduino-memory-flash-sram-eeprom/ (дата звернення: 14.05.2026).

6. What exactly does the term "shield" mean?. Arduino Forum. URL: https://forum.arduino.cc/t/what-exactly-does-the-term-shield-mean/13807 (дата звернення: 15.05.2026).

7. Karl Söderby & Jacob Hylén. Getting Started with Arduino IDE 2. Arduino Docs. URL: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/ (дата звернення: 15.05.2026).

8. Evans, Brian. Beginning arduino programming. Apress, 2011.

9. Yacoub, Michel Daoud. Wireless technology: protocols, standards, and techniques. CRC press. 2017.

10. Gordon ColbachGordon Colbach. The Wi-Fi Networking Book: WLAN Standards: IEEE 802.11 bgn, 802.11n, 802.11ac and 802.11a. 2019.

11. Koulouras, Grigorios, Stylianos Katsoulis, and Fotios Zantalis. Evolution of Bluetooth technology: BLE in the IoT ecosystem. Sensors 25.4. 2025, URL: https://www.mdpi.com/1424-8220/25/4/996 (дата звернення: 15.05.2026).

12. Dian, F. John, Amirhossein Yousefi, and Sungjoon Lim. A practical study on Bluetooth Low Energy (BLE) throughput. 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). IEEE, 2018. URL: https://ieeexplore.ieee.org/abstract/document/8614763 (дата звернення: 15.05.2026).

13. Ayoub Kamal, Muhammad, Alam, Muhammad Mansoor, Sajak, Aznida Abu Bakar, Mohd Su'ud, Mazliham. Requirements, Deployments, and Challenges of LoRa Technology: A Survey, Computational Intelligence and Neuroscience. C. 15. 2023. URL: https://doi.org/10.1155/2023/5183062 (дата звернення: 15.05.2026).

14. Damra, Hamza, Iacovos Ioannou, and Khalid A. Darabkh. IoT Mesh Networking: Performance Comparison of Z-Wave, Zigbee, BLE, and Thread with a Generic Mathematical Model. URL: https://www.researchgate.net/profile/Iacovos-Ioannou/publication/399666694 (дата звернення: 15.05.2026).

15. Gaikwad, Vijay, et al. Intelligent Traffic Management System using RFID and IR Sensors based on Arduino Uno. International Conference on Computer Science and Communication Engineering (ICCSCE 2025). Atlantis Press. 2025. URL: https://www.atlantis-press.com/proceedings/iccsce-25/126017419 (дата звернення: 15.05.2026).

16. Irving, Philip, and Pascal A. Ochang. Evolutionary analysis of GSM, UMTS and LTE mobile network architectures. World Scientific News 54. 2016. C. 27-39. URL: http://sure.sunderland.ac.uk/id/eprint/7512/ (дата звернення: 15.05.2026).

17. Oustry, Antoine, et al. Optimization in wireless networks. Encyclopedia of Optimization. Cham: Springer International Publishing. 2022. C. 1-11. URL: https://www.researchgate.net/publication/366634966_Optimization_in_Wireless_Networks (дата звернення: 16.05.2026).

18. Priyadarshi, R., Kumar, R.R., Ranjan, R. et al. AI-based routing algorithms improve energy efficiency, latency, and data reliability in wireless sensor networks. Sci Rep 15, 22292. 2025. URL: https://doi.org/10.1038/s41598-025-08677-w (дата звернення: 16.05.2026).

19. Pathan, Al-Sakib Khan, Hyung-Woo Lee, and Choong Seon Hong. "Security in wireless sensor networks: issues and challenges." 2006 8th International Conference Advanced Communication Technology. Vol. 2. IEEE. 2006. URL: https://ieeexplore.ieee.org/abstract/document/1625756 (дата звернення: 16.05.2026).

20. Soni, Dipa, and Ashwin Makwana. A survey on mqtt: a protocol of internet of things (iot). International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017). Vol. 20. No. April 2017. URL: https://www.researchgate.net/profile/Dipa-Soni/publication/316018571 (дата звернення: 16.05.2026).

21. Bormann, Carsten, Angelo P. Castellani, and Zach Shelby. Coap: An application protocol for billions of tiny internet nodes. IEEE Internet Computing 16.2. 2012. С. 62-67. URL: https://ieeexplore.ieee.org/abstract/document/6159216 (дата звернення: 17.05.2026).

22. Anthony Steed, Manuel Fradinho Oliveira. Networked Graphics. 2009. URL: https://doi.org/10.1016/B978-0-12-374423-4.00004-5 (дата звернення: 17.05.2026).

23. Arora, Mohit. Handling Endianness. The Art of Hardware Architecture: Design Methods and Techniques for Digital Circuits. New York, NY: Springer New York, 2011. С. 155-168. URL: https://link.springer.com/chapter/10.1007/978-1-4614-0397-5_7 (дата звернення: 17.05.2026).

24. Kumar, Natesh, et al. Evaluating Binary Serialization Protocols for IoT/M2M Applications over Hybrid Terrestrial and Non-Terrestrial Networks. Telecom. Vol. 7. No. 2. MDPI. 2026. URL: https://doi.org/10.3390/telecom7020043 (дата звернення: 18.05.2026).

25. Friesel, Daniel, and Olaf Spinczyk. Data Serialization Formats for the Internet of Things. Electronic Communications of the EASST, 2021. URL: https://www.eceasst.org/index.php/eceasst/article/view/2256 (дата звернення: 18.05.2026).

26. Z. Xu, Huihui, and Xiaobing Yuan. Cross-layer design for energy-efficient reliable multi-path transmission in event-driven wireless sensor networks. Sensors 23.14. 2023. URL: https://doi.org/10.3390/s23146520 (дата звернення: 18.05.2026).

27. Kumar, Rakesh, Diwakar Bhardwaj, and Manas Kumar Mishra. Event driven application specific prioritized data transmission approach for underwater wireless sensor networks. Journal of Scientific & Industrial Research 80. 2021. C. 276-282.

28. M. A. et al. A Comprehensive Review of Edge Computing: A Perspective of IoT, ResearchGate, 2026. URL: https://doi.org/10.64886/oajea.0101.004 (дата звернення: 18.05.2026).

29. Khairnar, Sushil. EDGE COMPUTING For IOT DEVICES: A COMPREHENSIVE FRAME FOR DISTRIBUTIVE DATA PROCESSING AND REAL-TIME ANALYTICS. Journal of Integrated Research 2.1. 2021. C. 1-14. URL: https://interresearcher.com/index.php/JIR/article/view/62 (дата звернення: 18.05.2026).

30. "AWS IoT". Amazon Web Services. URL: https://aws.amazon.com/iot/ (дата звернення: 20.06.2026).

31. Building an AWS IoT Core device using AWS Serverless and an ESP32. AWS Compute Blog. URL: https://aws.amazon.com/blogs/compute/building-an-aws-iot-core-device-using-aws-serverless-and-an-esp32/ (дата звернення: 20.06.2026).

32. Blynk: Low-Code IoT Platform. Blynk. URL: https://www.blynk.io/ (дата звернення: 20.06.2026).

33. Arduino Cloud | Build, Control, Monitor Your IoT Projects. Arduino. URL: https://cloud.arduino.cc/ (дата звернення: 20.06.2026).

34. MQTT - Home Assistant. Home Assistant. URL: https://www.home-assistant.io/integrations/mqtt/ (дата звернення: 20.06.2026).

35. Working with MQTT in Node-RED: Complete Guide. FlowFuse. URL: https://flowfuse.com/blog/2024/06/how-to-use-mqtt-in-node-red/ (дата звернення: 20.06.2026).

36. Barot V, Kapadia V, Pandya S. QoS enabled IoT based low cost air quality monitoring system with power consumption optimization. Cybernetics and Information Technologies. 2020;20(2):122-40. URL: https://reference-global.com/download/article/10.2478/cait-2020-0021.pdf (дата звернення: 21.05.2026).

37. Diaz K, Teh YK. Power Optimization of Arduino-Based Sensor System for Salton Sea Environmental Monitoring. In2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS). 2019. С. 1215-1218. URL: https://ieeexplore.ieee.org/abstract/document/8884880 (дата звернення: 21.05.2026).

38. Zain-Aldeen S. A.Rhman, Ramzy S. Ali, Basil H. Jasim. Wirelessly controlled irrigation system. Dept. of Electrical Engineering University of Basrah. 2014. URL: https://doi.org/10.37917/ijeee.10.2.4 (дата звернення: 21.05.2026).

39. Masykuroh K, Syifa FT, Setiyanto GR, Ramadhani AD, Kurnianto D, Iryani N. Prototype smart door lock by using wireless network based on arduino uno. In2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT). 2021. С. 342-347. URL: https://ieeexplore.ieee.org/abstract/document/9530806 (дата звернення: 21.05.2026).

40. Sari EU, Setyawan TA, Nursyahid A, Enriko KA, Widodo S. Automatic Control of Hydroponic Nutrient Solution Concentration Based on Edge and Cloud Computing Using Message Queuing Telemetry Transport (MQTT) Protocol. In2021 8th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE). 2021. С. 207-212. URL:

https://ieeexplore.ieee.org/abstract/document/9617513 (дата звернення: 21.05.2026).

41. Murugan K, Alaguselvi R. Arduino Based Wireless Sensor Network Systems for Agriculture. 3C Tecnología. Glosas de innovación aplicadas a la pyme, Edición Especial. 2021. С. 87-97. URL: https://pdfs.semanticscholar.org/7835/fd4e709eb561a1dfa6618d95c06dd490e644.pdf (дата звернення: 21.05.2026).

42. Math A, Ali L, Pruthviraj U. Development of smart drip irrigation system using IoT. In2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER). 2018. С. 126-130. URL: https://ieeexplore.ieee.org/abstract/document/8674080 (дата звернення: 21.05.2026).

43. Sun Z, Dyke SJ, Pena F, Wilbee A. Development of Arduino based wireless control system. InSensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems. 2015. С. 416-426. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/9435/94351D/Development-of-Arduino-based-wireless-control-ystem/10.1117/12.2083707.short (дата звернення: 21.05.2026).