БАКАЛАВРСЬКА РОБОТА

КРБ.СІ-10.00.00.000 ПЗ

Група СІ-22-1

Олександр ТИМОФІЙЧУК

2026

Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інформаційно-телекомунікаційних технологій та систем

Тимофійчук Олександр Сергійович

(прізвище, ім'я, по батькові)

УДК 621.396.931:004.7

БАКАЛАВРСЬКА РОБОТА

Розроблення системи телеметрії на базі Bluetooth-модуля CC2541

(назва роботи)

Системна інженерія – інтернет речей

(назва освітньої програми)

151-Автоматизація та комп'ютерно-інтегровані технології

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____ О.С. Тимофійчук
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____ Еліяшів Олег Миронович, к.т.н.
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
В.о. завідувача кафедри

_____ ***Заміховська О.Л.***
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Івано-Франківський національний технічний університет нафти і газу

(повне найменування закладу вищої освіти)

Факультет *Інформаційних технологій*

Кафедра *Інформаційно-телекомунікаційних технологій та систем*

Освітній рівень *бакалавр*

Спеціальність *151-Автоматизація та комп'ютерно-інтегровані технології*

(шифр і назва)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри *ІТТС*

к.т.н.,доцент.

О.Л. Заміховська

«____» 2026 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Тимофійчуку Олександру Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Розроблення системи телеметрії на базі Bluetooth-модуля CC2541*

керівник роботи *Еліяшів Олег Миронович, к.т.н.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від "*7*" *квітня* 2026 року № *164/7*

2. Строк подання студентом роботи

3. Вихідні дані до роботи *Матеріали та результати отримані під час проходження переддипломної практики, технічні вимоги, методичні вказівки.*

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

Порівняльний аналіз існуючих систем телеметрії

Вибір апаратних технологій та середовища розробки

Розробка апаратної частини

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

Структурна схема

Функціональна схема

Результати розробки

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	*Назва етапів бакалаврської роботи*	*Термін виконання Етапів роботи*	*Примітка*
1	*Порівняльний аналіз існуючих систем телеметрії*	*02.03.2026-25.03.2026*	*Виконано*
2	*Вибір середовища розробки і основних вузлів управління*	*04.04.2026-15.04.2026*	*Виконано*
3	*Розробка апаратної частини і схеми управління*	*20.04.2026-01.05.2026*	*Виконано*
4	*Оформлення результатів роботи*	*13.05.2026-02.06.2026*	*Виконано*

Студент _____ *Тимофійчук О.С.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Еліяшів О.М.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Розрахунково-пояснювальна записка: 108 сторінок, 36 рисунків, 3 таблиці, 25 посилань.

Об'єктом дослідження є система телеметрії на базі Bluetooth-модуля CC2541, що забезпечує бездротову передачу телеметричних даних за технологією Bluetooth Low Energy 4.0.

Мета роботи – розроблення системи телеметрії на базі BLE-модуля CC2541, що включає апаратний вимірювальний вузол, протокол передачі даних та програмне забезпечення приймальної сторони для персонального комп'ютера та мобільного пристрою.

У першому розділі проведено аналіз предметної області: розглянуто поняття телеметрії та її застосування в автоматизованих системах, виконано порівняльний аналіз технологій бездротової передачі даних. та виконано порівняльний аналіз існуючих рішень на базі BLE-модулів.

У другому розділі здійснено розроблення апаратного та програмного забезпечення системи телеметрії. Розглянуто архітектуру та технічні характеристики мікроконтролера CC2541, будову та розпіновку модуля HM-10, інтерфейс UART та систему AT-команд. Виконано вибір та обґрунтування датчиків: DS18B20, MPX5700, ADXL345, INA219. Розроблено функціональну схему системи, розроблено програмне забезпечення мікроконтролерного вузлу

У третьому розділі проведено комплексне тестування системи відповідно до розробленої методики. Виконано перевірку датчиків, AT-команд модуля HM-10 та BLE-з'єднання у трьох сценаріях. Проведено вимірювання дальності BLE-з'єднання на відстанях від 5 до 40 метрів у приміщенні та надворі, а також аналіз отриманих результатів і порівняння з вимогами технічного завдання.

ТЕЛЕМЕТРІЯ, BLUETOOTH LOW ENERGY, CC2541, HM-10, ARDUINO, БЕЗДРОТОВА ПЕРЕДАЧА ДАНИХ, ДАТЧИКИ, ПРОТОКОЛ ПЕРЕДАЧІ, ПРОГРАМА-КЛІЄНТ.

ABSTRACT

Calculation and explanatory note: 108pages, 36 figures, 3 tables, 25 references.

The object of the study is a telemetry system based on the Bluetooth module CC2541, which provides wireless transmission of telemetric data using Bluetooth Low Energy 4.0 technology.

The purpose of the work is to develop a telemetry system based on the BLE module CC2541, which includes a hardware measuring node, a data transmission protocol and receiving side software for a personal computer and a mobile device.

The first section analyzes the subject area: the concept of telemetry and its application in automated systems are considered, a comparative analysis of wireless data transmission technologies is performed. and a comparative analysis of existing solutions based on BLE modules is performed.

The second section develops the hardware and software of the telemetry system. The architecture and technical characteristics of the CC2541 microcontroller, the structure and pinout of the HM-10 module, the UART interface and the AT-command system are considered. The selection and justification of the sensors: DS18B20, MPX5700, ADXL345, INA219 are performed. The functional diagram of the system is developed, the software of the microcontroller node is developed

In the third section, a comprehensive testing of the system is carried out in accordance with the developed methodology. The sensors, AT-commands of the HM-10 module and BLE-connection are checked in three scenarios. The BLE-connection range is measured at distances from 5 to 40 meters indoors and outdoors, as well as the analysis of the results obtained and comparison with the requirements of the technical specifications.

TELEMETRY, BLUETOOTH LOW ENERGY, CC2541, HM-10, ARDUINO, WIRELESS DATA TRANSMISSION, SENSORS, TRANSMISSION PROTOCOL, CLIENT PROGRAM.

ЗМІСТ

с.

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

АСКТП — автоматизована система керування технологічним процесом

САК — система автоматичного керування

ККД — коефіцієнт корисної дії

ПЛК — програмований логічний контролер

МК — мікроконтролер

ВСТУП

Сучасний розвиток промислової автоматизації, робототехніки та комп'ютерно-інтегрованих технологій висуває дедалі вищі вимоги до систем збору, передачі та обробки даних у реальному часі. Телеметрія — як технологія дистанційного вимірювання фізичних параметрів об'єктів та передачі отриманих даних на відстань — є невід'ємною складовою сучасних автоматизованих систем керування. Вона знаходить широке застосування в промисловості, медицині, транспорті, сільському господарстві, системах розумного будинку та інтернеті речей (IoT). Ключовою вимогою до таких систем є надійність, енергоефективність та можливість бездротової передачі даних без прив'язки до провідної інфраструктури. Традиційні провідні рішення, хоча і забезпечують високу стабільність каналу зв'язку, суттєво обмежують мобільність об'єктів моніторингу, підвищують вартість монтажу та обслуговування, а також ускладнюють масштабування системи. Саме тому бездротові технології передачі даних стають дедалі більш затребуваними як в академічному середовищі, так і в промисловому секторі.

Серед сучасних технологій бездротового зв'язку особливе місце займає Bluetooth Low Energy (BLE) — стандарт, що був введений у специфікацію Bluetooth 4.0 і орієнтований саме на застосування з низьким енергоспоживанням. На відміну від класичного Bluetooth, технологія BLE дозволяє пристроям працювати від автономних джерел живлення протягом тривалого часу — від кількох місяців до кількох років залежно від інтенсивності передачі даних, — що є критично важливим для мобільних та вбудованих систем телеметрії. Саме ця властивість зумовила стрімке поширення BLE у галузях промислової автоматизації, медичного моніторингу, розумних будинків та носимої електроніки. Порівняно з іншими бездротовими технологіями, такими як Wi-Fi, ZigBee або LoRa, протокол BLE вирізняється

оптимальним поєднанням низького енергоспоживання, достатньої пропускної здатності для передачі телеметричних даних, широкою підтримкою з боку сучасних смартфонів та операційних систем, а також відносно невисокою вартістю апаратної реалізації. Ці переваги роблять BLE одним із найбільш перспективних стандартів для побудови систем бездротової телеметрії малого та середнього радіусу дії.

Одним із широко застосовуваних апаратних рішень для побудови BLE-систем є мікроконтролер CC2541 виробництва Texas Instruments — спеціалізований однокристальний пристрій, що поєднує в собі мікроконтролерне ядро на базі архітектури 8051 та повнофункціональний радіомодуль стандарту BLE 4.0. На базі цього мікроконтролера створено значну кількість готових модулів, зокрема широко відомий модуль HM-10 китайської компанії Jinan Huamao Technology. Цей модуль реалізує повний BLE-стек на апаратному рівні та надає розробнику зручний інтерфейс керування через AT-команди по інтерфейсу UART, що суттєво спрощує розробку кінцевих пристроїв і скорочує час виходу продукту на ринок. Завдяки такому підходу розробник може зосередитись на логіці роботи кінцевого пристрою, не заглиблюючись у деталі реалізації BLE-стеку на низькому рівні. Модуль підтримує режими ведучого (Central) та веденого (Peripheral) пристрою, що дозволяє будувати як прості топології «точка — точка», так і більш складні мережі з кількох вузлів. Напруга живлення модуля складає від 3,0 до 5,0 В, струм споживання в активному режимі — близько 9 мА, а в режимі сну — від 50 до 200 мкА, що підтверджує його придатність для автономних рішень.

Актуальність теми бакалаврської роботи обумовлена зростаючим попитом на компактні, енергоефективні та доступні за вартістю системи бездротової телеметрії для задач автоматизації та моніторингу. У сучасних умовах розвитку концепції Індустрії 4.0 та інтернету речей системи телеметрії набувають стратегічного значення як інструмент забезпечення прозорості

виробничих процесів, своєчасного виявлення відхилень від норми та прийняття управлінських рішень на основі актуальних даних. Незважаючи на широкий вибір готових комерційних рішень, значна частина промислових і дослідницьких задач вимагає розроблення власних спеціалізованих систем, адаптованих під конкретні умови експлуатації, набір вимірюваних параметрів та вимоги до інтерфейсу користувача. Розроблення системи телеметрії на базі Bluetooth-модуля CC2541 є практично значущим завданням, що поєднує в собі проектування апаратної частини, розробку вбудованого програмного забезпечення та створення клієнтського програмного засобу для прийому й відображення телеметричних даних. Крім того, дана тема має чітку прикладну спрямованість і може слугувати основою для подальших досліджень у галузі промислового IoT та розподілених систем моніторингу.

Метою бакалаврської роботи є розроблення функціональної системи телеметрії на базі Bluetooth-модуля CC2541, що забезпечує збір даних з підключених датчиків, їх бездротову передачу за технологією BLE та відображення на персональному комп'ютері або мобільному пристрої.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати існуючі технології бездротової передачі даних та обґрунтувати вибір BLE як основи для системи телеметрії;

- вивчити архітектуру мікроконтролера CC2541 та технічні характеристики модуля HM-10;

- розробити апаратну схему системи телеметрії з підключенням датчиків до мікроконтролерного вузлу;

- розробити програмне забезпечення мікроконтролерного вузлу для збору, первинної обробки та передачі телеметричних даних через BLE-канал;

- розробити програму-клієнт для персонального комп'ютера та/або мобільного пристрою для прийому і відображення даних;

- провести тестування системи та оцінити її основні характеристики: дальність зв'язку, стабільність з'єднання та енергоспоживання.

Об'єктом дослідження є процес бездротової передачі телеметричних даних у системах автоматизації на основі технології Bluetooth Low Energy.

Предметом дослідження є система телеметрії, побудована на базі Bluetooth-модуля CC2541 з використанням інтерфейсу UART та протоколу BLE 4.0.

Методи дослідження. У роботі використано методи системного аналізу при проектуванні архітектури системи, методи схемотехнічного проектування при розробці апаратної частини, методи об'єктно-орієнтованого та процедурного програмування при розробці програмного забезпечення, а також експериментальні методи при тестуванні та оцінці характеристик розробленої системи. Під час аналізу предметної області застосовувався метод порівняльного аналізу для вибору оптимальної бездротової технології та елементної бази.

Практичне значення роботи полягає в тому, що розроблена система телеметрії може бути використана для моніторингу фізичних параметрів у лабораторних умовах, на виробництві, у складі мобільних роботизованих платформ, а також як основа для подальшого розширення функціональності у рамках більш складних автоматизованих комплексів. Розроблені програмні та апаратні рішення є відтворюваними і можуть бути адаптовані для широкого кола практичних задач у галузі автоматизації та робототехніки. Отримані результати тестування дозволяють оцінити застосовність модуля CC2541 для задач промислової телеметрії та визначити межі його ефективного використання в реальних умовах експлуатації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття телеметрії та її застосування в автоматизованих системах

Телеметрія (від грецьких слів «tele» - далеко та «metron» - вимірювання) – це технологія автоматизованого збору вимірювальної інформації про стан об'єкта або процесу та її передачі на відстань до пункту прийому й обробки. Поняття телеметрії охоплює весь ланцюжок перетворень - від первинного вимірювання фізичної величини датчиком до представлення результату у зручному для аналізу вигляді на стороні споживача інформації. Таким чином, телеметрична система є комплексним рішенням, що включає в себе вимірювальні перетворювачі, засоби первинної обробки сигналів, канал передачі даних та програмне забезпечення для прийому, зберігання і відображення телеметричної інформації [1,2].

Історично поняття телеметрії виникло у зв'язку з потребою здійснювати вимірювання на об'єктах, доступ до яких був фізично ускладнений або неможливий. Перші телеметричні системи з'явились у кінці XIX – на початку XX століття і використовувались переважно в метеорології та енергетиці. Значного поштовху розвиток телеметрії набув у середині XX століття в рамках космічних програм, коли виникла необхідність отримувати в реальному часі дані про стан бортових систем ракет та супутників. Саме космічна галузь сформувала основні вимоги до телеметричних систем: висока надійність, мінімальне енергоспоживання, завадостійкість каналу зв'язку та здатність до роботи в умовах динамічного середовища. Згодом ці принципи були перенесені в промисловість, транспорт, медицину та багато інших галузей.

У сучасному розумінні телеметрія є невід'ємною складовою автоматизованих систем керування технологічними процесами (АСКТП),

систем диспетчерського керування та збору даних (SCADA), а також розподілених систем моніторингу на базі концепції інтернету речей (IoT). Вона дозволяє в режимі реального часу отримувати актуальну інформацію про стан обладнання, технологічних параметрів або навколишнього середовища, що є необхідною умовою для прийняття своєчасних і обґрунтованих управлінських рішень. Відсутність або затримка телеметричної інформації в автоматизованих системах може призводити до виникнення аварійних ситуацій, зниження якості продукції або суттєвих економічних збитків [3].

Структурно будь-яка система телеметрії складається з трьох основних функціональних блоків. Перший блок – вимірювальний, що включає датчики фізичних величин (температури, тиску, вологості, прискорення, напруги, струму тощо) та аналого-цифрові перетворювачі для оцифрування аналогових сигналів. Другий блок – передавальний, що забезпечує формування пакетів даних та їх передачу по каналу зв'язку – провідному або бездротовому. Третій блок – приймальний, що здійснює прийом, декодування, зберігання та відображення телеметричних даних у зручному для оператора або автоматизованої системи вигляді. У сучасних реалізаціях ці блоки можуть бути реалізовані як на окремих апаратних платформах, так і інтегровані в єдиний мікроконтролерний вузол.

Застосування телеметричних систем у галузі промислової автоматизації є надзвичайно широким. У нафтогазовій промисловості телеметрія використовується для моніторингу стану свердловин, трубопроводів та насосних станцій у режимі реального часу. В енергетиці телеметричні системи забезпечують збір даних з розподілених підстанцій, лічильників електроенергії та елементів розумних електромереж (Smart Grid). У виробничих процесах телеметрія дозволяє контролювати параметри технологічних ліній, своєчасно виявляти відхилення від норми та запобігати поломкам обладнання завдяки системам прогностичного технічного обслуговування. Широко застосовується телеметрія і в робототехніці – для передачі даних про стан приводів, датчиків

та навігаційних систем мобільних роботів до пульта оператора або центральної системи керування.

У медицині телеметричні системи застосовуються для дистанційного моніторингу стану пацієнтів – реєстрації електрокардіограми, рівня насичення крові киснем, артеріального тиску та інших фізіологічних параметрів без прив'язки пацієнта до стаціонарного обладнання. Це дозволяє суттєво підвищити комфорт пацієнта та якість медичної допомоги, а також знизити навантаження на медичний персонал. У сільському господарстві системи телеметрії використовуються для моніторингу мікроклімату в теплицях, вологості ґрунту на полях, стану сільськогосподарської техніки та умов зберігання зерна. Телематичні системи на транспорті дозволяють відстежувати місцезнаходження, швидкість, витрату пального та технічний стан транспортних засобів у режимі реального часу, що є основою сучасних систем управління автопарком та логістикою.

Важливою тенденцією останніх років є інтеграція телеметричних систем у концепцію інтернету речей та промислового інтернету речей (IIoT). В рамках цієї концепції окремі телеметричні вузли об'єднуються у розподілені мережі, де кожен пристрій може як збирати дані, так і передавати їх до хмарних платформ для подальшого аналізу з використанням методів машинного навчання та великих даних. Бездротові технології передачі даних відіграють ключову роль у побудові таких мереж, оскільки дозволяють підключати до системи моніторингу об'єкти, розташовані у важкодоступних місцях або такі, що постійно переміщуються в просторі.

Таким чином, телеметрія є фундаментальною технологією сучасної автоматизації, що забезпечує інформаційний зв'язок між фізичним об'єктом керування та системою прийняття рішень. Ефективність автоматизованої системи в цілому значною мірою визначається якістю та своєчасністю телеметричної інформації, що надходить від польових пристроїв. Саме тому розроблення надійних, економічних та функціонально розвинених систем

телеметрії залишається актуальним науково-технічним завданням, а вибір оптимальної технології бездротової передачі даних є одним із ключових етапів проектування таких систем.

1.2 Огляд технологій бездротової передачі даних для систем телеметрії

Вибір технології бездротової передачі даних є одним із найважливіших етапів проектування системи телеметрії, оскільки саме від цього вибору залежать ключові характеристики всієї системи – дальність дії, енергоспоживання, пропускна здатність каналу зв'язку, вартість апаратної реалізації та сумісність з існуючою інфраструктурою. На сьогоднішній день існує значна кількість стандартів і технологій бездротового зв'язку, кожна з яких має свої переваги та обмеження і орієнтована на певний клас застосувань. Для обґрунтованого вибору оптимальної технології необхідно провести порівняльний аналіз основних представників цього класу рішень.

Wi-Fi (IEEE 802.11)

Технологія Wi-Fi є одним із найбільш поширених стандартів бездротового зв'язку у світі. Вона забезпечує високу пропускну здатність каналу – від десятків до сотень мегабіт за секунду залежно від версії стандарту та працює в діапазонах частот 2,4 ГГц і 5 ГГц. Дальність дії у відкритому просторі може досягати кількох сотень метрів, а в приміщенні – як правило, від 30 до 100 метрів. Завдяки широкій поширеності Wi-Fi-інфраструктури інтеграція телеметричного пристрою до існуючої мережі є відносно простою задачею [4, 5].

Однак для систем телеметрії з автономним живленням технологія Wi-Fi має суттєвий недолік – значне енергоспоживання. Струм споживання Wi-Fi-модуля в активному режимі може становити від 100 до 300 мА і більше, що

робить неможливим тривалу автономну роботу від батарейного джерела живлення. Крім того, Wi-Fi потребує наявності точки доступу та відповідної мережевої інфраструктури, що не завжди є можливим в польових умовах. Тому Wi-Fi доцільно застосовувати переважно у стаціонарних системах телеметрії з постійним живленням від мережі та за наявності готової Wi-Fi-інфраструктури.

ZigBee (IEEE 802.15.4)

ZigBee – це стандарт бездротового зв'язку, розроблений спеціально для застосувань з низьким енергоспоживанням та невисокими вимогами до пропускної здатності. Він працює в діапазоні частот 2,4 ГГц (а також 868 МГц і 915 МГц у регіональних варіантах) та забезпечує швидкість передачі даних до 250 кбіт/с. Дальність дії одного вузла складає від 10 до 100 метрів, проте завдяки підтримці мережевої топології типу «mesh» загальне покриття мережі ZigBee може бути значно більшим – окремі вузли ретранслюють сигнал, збільшуючи радіус мережі до кількох кілометрів.

Енергоспоживання пристроїв ZigBee є значно нижчим порівняно з Wi-Fi – в режимі сну струм споживання може становити одиниці мікроампер, що дозволяє забезпечити роки автономної роботи від невеликої батареї. Саме тому ZigBee широко застосовується в системах автоматизації будівель, розумних будинках та промислових мережах датчиків. Разом з тим, технологія ZigBee має обмежену підтримку з боку споживчих пристроїв – зокрема, смартфони, як правило, не мають вбудованого ZigBee-адаптера, що ускладнює безпосередню взаємодію з мобільними пристроями без додаткового шлюзового обладнання [6, 7].

LoRa / LoRaWAN

LoRa (Long Range) – технологія бездротового зв'язку, що використовує метод модуляції з розширеним спектром (Chirp Spread Spectrum) та працює в неліцензованих діапазонах частот – 433 МГц, 868 МГц (Європа) та 915 МГц (США). Головною відмінністю LoRa від інших розглянутих технологій є

виняткова дальність зв'язку – в умовах прямої видимості вона може досягати 15–20 км, а в міській забудові – 2-5 км. При цьому енергоспоживання залишається відносно невисоким, що робить LoRa ідеальним вибором для систем телеметрії з великою кількістю розподілених вузлів на значних відстанях.

Проте технологія LoRa має суттєве обмеження – дуже низьку пропускну здатність каналу, що становить від 0,3 до 50 кбіт/с залежно від налаштувань. Це робить LoRa непридатною для передачі великих обсягів даних або потокового відео, однак цілком достатньою для передачі коротких пакетів телеметричної інформації. LoRaWAN як протокол мережевого рівня над LoRa додає можливості адресації, шифрування та підтримки великої кількості кінцевих пристроїв в одній мережі. Застосування LoRa є найбільш доцільним для систем моніторингу сільськогосподарських угідь, трубопровідних мереж, систем обліку комунальних ресурсів та інших застосувань, де потрібна велика дальність при мінімальному обсязі переданих даних [8].

GSM / LTE (стільниковий зв'язок)

Використання стільникових мереж для передачі телеметричних даних є поширеним рішенням у випадках, коли необхідно забезпечити зв'язок на великих відстанях або в умовах відсутності локальної бездротової інфраструктури. Модулі GSM/GPRS або LTE дозволяють передавати дані через загальнодоступну мережу мобільного зв'язку практично з будь-якої точки, де є покриття оператора. Такі рішення широко застосовуються в системах моніторингу транспортних засобів, автоматизованих системах обліку електроенергії, газу та води, а також у системах охоронної сигналізації.

Основними недоліками GSM/LTE-рішень є відносно високе енергоспоживання, необхідність SIM-карти та оплати послуг, залежність від покриття стільникової мережі. Тому стільниковий зв'язок є доцільним для систем з постійним або достатньо потужним джерелом живлення та там, де інші технології не забезпечують необхідної дальності зв'язку.

Bluetooth та Bluetooth Low Energy (BLE)

Класичний Bluetooth (версії до 3.0) забезпечує швидкість передачі даних до 3 Мбіт/с на відстані до 100 метрів, однак має відносно високе енергоспоживання та не є оптимальним для систем телеметрії з автономним живленням. Принципово нову можливість відкрила специфікація Bluetooth 4.0, що включила стандарт Bluetooth Low Energy – технологію, спеціально оптимізовану для застосувань з мінімальним енергоспоживанням. BLE забезпечує передачу даних зі швидкістю до 1 Мбіт/с на відстані до 100 метрів у відкритому просторі при струмі споживання в режимі передачі близько 9-15 мА та в режимі сну – менше 1 мкА [7,8].

Ключовою перевагою BLE над іншими розглянутими технологіями є його підтримка практично всіма сучасними смартфонами та планшетами під управлінням Android та iOS, що дозволяє використовувати мобільні пристрої як зручний інтерфейс для відображення телеметричних даних без додаткового обладнання. Це суттєво знижує загальну вартість системи та спрощує її розгортання. Крім того, BLE підтримує шифрування даних на апаратному рівні, що забезпечує необхідний рівень інформаційної безпеки .

Для наочного представлення результатів порівняльного аналізу розглянутих технологій наведемо їх основні характеристики у вигляді таблиці.

Таблиця 1.1 – Порівняльна характеристика технологій бездротової передачі даних

Технологія	**Дальність**	**Швидкість передачі**	**Енергоспо-живання**	**Підтримка смартфонів**	**Вартість модуля**
Wi-Fi	до 300 м	до 600 Мбіт/с	Високе	Так	Середня
ZigBee	до 100 м (mesh - км)	до 250 кбіт/с	Низьке	Ні	Середня
LoRa	до 20 км	до 50 кбіт/с	Низьке	Ні	Середня
GSM/LTE	необмежена	до 100 Мбіт/с	Високе	Так	Висока
BLE 4.0	до 100 м	до 1 Мбіт/с	Дуже низьке	Так	Низька

За результатами порівняльного аналізу можна зробити висновок, що для розроблення системи телеметрії малого радіусу дії з автономним живленням та можливістю взаємодії зі смартфоном технологія Bluetooth Low Energy є оптимальним вибором. Вона забезпечує достатню дальність зв'язку для більшості лабораторних та виробничих застосувань, мінімальне енергоспоживання, широку підтримку з боку споживчих пристроїв та низьку вартість апаратної реалізації. Саме тому в даній бакалаврській роботі як основу системи телеметрії обрано Bluetooth-модуль на базі мікроконтролера CC2541, що реалізує стандарт BLE 4.0.

1.3 Протокол Bluetooth Low Energy (BLE 4.0): архітектура та принципи роботи

Bluetooth Low Energy (BLE) — це стандарт бездротового зв'язку малого радіусу дії, що був введений у складі специфікації Bluetooth 4.0, опублікованої організацією Bluetooth Special Interest Group (Bluetooth SIG) у 2010 році. На відміну від класичного Bluetooth, який орієнтований на передачу відносно великих обсягів даних (аудіо, файли), BLE з самого початку проектувався як технологія для пристроїв з екстремально низьким енергоспоживанням, що передають невеликі пакети даних з певною періодичністю. Саме ця концепція визначила архітектурні рішення протоколу та зумовила його широке застосування в системах телеметрії, медичних пристроях, носимій електроніці та промисловому інтернеті речей [9, 10].

Важливо розуміти, що BLE та класичний Bluetooth є двома різними протоколами в рамках єдиного стандарту Bluetooth 4.0. Вони використовують однаковий радіодіапазон - 2,4 ГГц, але несумісні між собою на рівні протоколу. Пристрої, що підтримують обидва режими, називаються dual-mode і здатні одночасно підтримувати з'єднання як по класичному Bluetooth, так і по

BLE. Пристрої, що підтримують лише BLE, називаються single-mode і є типовими для вбудованих систем та датчиків з батарейним живленням.

Фізичний рівень BLE

На фізичному рівні BLE використовує діапазон частот 2,4 - 2,4835 ГГц, розділений на 40 радіоканалів з кроком 2 МГц. З них 37 каналів використовуються для передачі даних, а 3 канали – з номерами 37, 38 та 39 – призначені для широкомовної реклами (advertising), тобто для оголошення пристроєм своєї присутності та доступності для підключення, рис.1.1. Такий розподіл каналів дозволяє мінімізувати взаємні завади між процедурою виявлення пристроїв та передачею корисних даних [11].

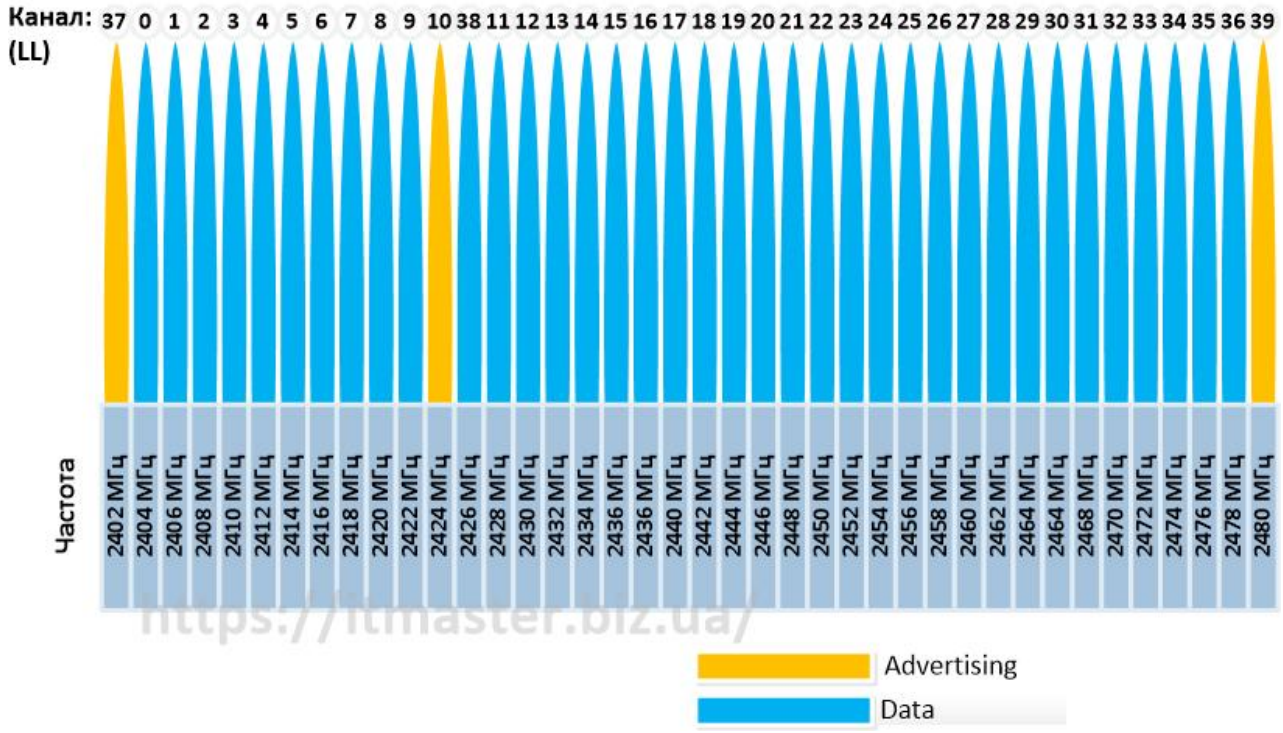


Рисунок 1.1 – Фізичний рівень BLE

Для модуляції сигналу BLE використовує метод гауссової частотної маніпуляції (GFSK - Gaussian Frequency Shift Keying) з індексом модуляції 0,5. Швидкість передачі даних на фізичному рівні становить 1 Мбіт/с у версії BLE 4.0, тоді як у пізніших версіях стандарту (BLE 5.0 і вище) була додана підтримка режиму 2 Мбіт/с. Для підвищення завадостійкості BLE

використовує технологію адаптивного стрибкоподібного перестроювання частоти (FHSS - Frequency Hopping Spread Spectrum), що дозволяє уникати зайнятих каналів та зменшувати вплив вузькосмугових завад.

Вихідна потужність передавача BLE відповідно до стандарту може варіюватись від -20 дБм до +10 дБм, що дозволяє гнучко регулювати компроміс між дальністю зв'язку та енергоспоживанням. Для модуля НМ-10 на базі CC2541 доступні значення вихідної потужності -23 дБм, -6 дБм, 0 дБм та +6 дБм, що налаштовуються програмно через відповідну АТ-команду [12,13].

Стек протоколів BLE

Архітектура програмного забезпечення BLE організована у вигляді багаторівневого стеку протоколів, який прийнято поділяти на дві основні частини – контролер (Controller) та хост (Host), між якими визначено стандартний інтерфейс HCI (Host Controller Interface), рис.1.2.

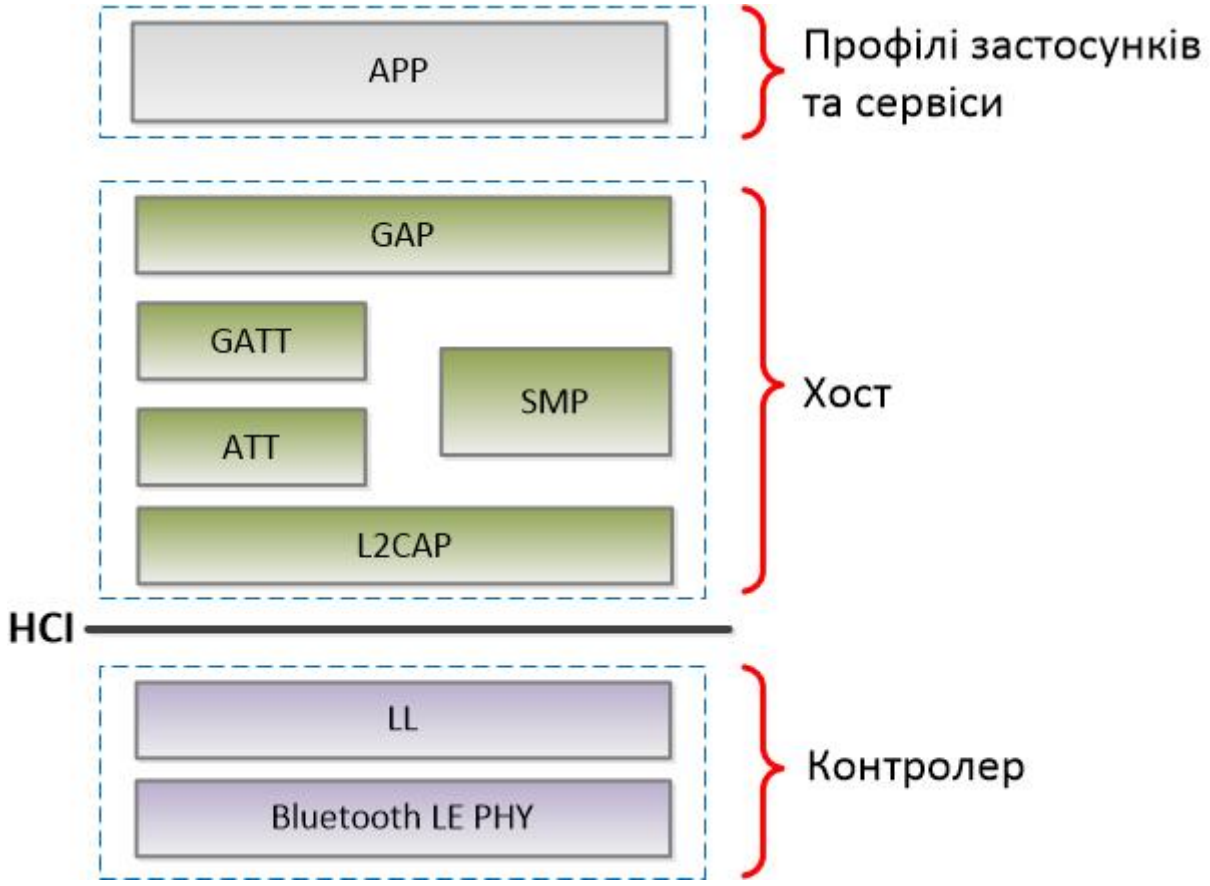


Рисунок 1.2 – LE стек BLE

Контролер включає фізичний рівень (PHY) та рівень управління доступом до середовища (Link Layer). Рівень Link Layer відповідає за формування пакетів даних, управління з'єднанням, шифрування на апаратному рівні та реалізацію процедур рекламування і сканування. Саме на цьому рівні реалізується механізм стрибкоподібного перестроювання частоти та управління параметрами з'єднання.

Хост включає кілька рівнів вищого порядку. Рівень L2CAP (Logical Link Control and Adaptation Protocol) забезпечує мультиплексування даних від різних протоколів верхнього рівня та фрагментацію великих пакетів. Рівень SM (Security Manager) відповідає за процедури аутентифікації та шифрування з'єднання, включаючи процедуру сполучення (pairing) пристроїв. Рівень GAP (Generic Access Profile) визначає загальні процедури виявлення пристроїв, встановлення з'єднання та управління режимами роботи. Рівень GATT (Generic Attribute Profile) є ключовим для обміну даними між пристроями і визначає модель організації даних у вигляді ієрархії сервісів та характеристик.

Модель даних GATT

Протокол GATT визначає стандартизовану модель організації та обміну даними між BLE-пристроями, що є основою для взаємосумісності пристроїв різних виробників. В рамках моделі GATT всі дані організовані у вигляді ієрархічної структури атрибутів. Кожен атрибут має унікальний ідентифікатор – UUID (Universally Unique Identifier), значення та набір дозволів на читання і запис.

Рисунок 1.3 – Протокол GATT

На верхньому рівні ієрархії знаходяться сервіси (Services) — логічні групи пов'язаних між собою характеристик. Наприклад, стандартний сервіс

«Heart Rate» об'єднує характеристики, пов'язані з вимірюванням частоти серцевих скорочень. Кожен сервіс містить одну або кілька характеристик (Characteristics) — це базові одиниці даних у моделі GATT. Характеристика має значення (value), набір властивостей (read, write, notify, indicate) та може містити дескриптори (Descriptors) з додатковою метаінформацією.

Bluetooth SIG визначає стандартні UUID для типових сервісів та характеристик, що забезпечує взаємосумісність пристроїв різних виробників. Для власних (кастомних) застосувань, таких як модуль HM-10, використовуються 128-бітні UUID, що генеруються виробником. Саме через кастомний сервіс з однією характеристикою на читання та запис модуль HM-10 реалізує прозорий канал передачі даних, що функціонує аналогічно послідовному порту — так звана концепція Serial Port Profile поверх BLE.

Ролі пристроїв у BLE

Стандарт BLE визначає кілька ролей, які можуть виконувати пристрої в рамках з'єднання. На рівні GAP пристрої поділяються на Peripheral (периферійний) та Central (центральний). Периферійний пристрій є, як правило, маломіцним датчиком або виконавчим пристроєм, що рекламує свою присутність та очікує підключення. Центральний пристрій — це зазвичай смартфон, планшет або шлюзовий пристрій, що сканує ефір, виявляє периферійні пристрої та ініціює з'єднання з ними.

На рівні GATT пристрої можуть виступати в ролях Server (сервер) та Client (клієнт). Сервер зберігає дані у вигляді атрибутів та надає до них доступ. Клієнт ініціює операції читання, запису та підписки на сповіщення. Як правило, периферійний пристрій виконує роль GATT-сервера, а центральний – роль GATT-клієнта, хоча стандарт допускає і зворотну конфігурацію. Модуль HM-10 за замовчуванням працює в режимі Peripheral/Slave, однак може бути перемкнутий у режим Central/Master за допомогою AT-команди AT+ROLE1, що дозволяє будувати системи з передачею даних між двома модулями без участі смартфона.

Процедура встановлення з'єднання

Процес встановлення BLE-з'єднання складається з кількох етапів. На першому етапі периферійний пристрій виконує процедуру рекламування – з певною періодичністю (advertising interval) він надсилає рекламні пакети на трьох виділених каналах (37, 38, 39). Ці пакети містять ідентифікатор пристрою, його ім'я, перелік підтримуваних сервісів та інші параметри. Інтервал рекламування може варіюватись від 20 мс до 10,24 с – чим більший інтервал, тим нижче енергоспоживання, але тим довше центральний пристрій витрачає часу на виявлення периферійного.

На другому етапі центральний пристрій виконує сканування ефіру, приймаючи рекламні пакети від доступних периферійних пристроїв. Після виявлення потрібного пристрою центральний ініціює процедуру підключення, надсилаючи запит на з'єднання з параметрами – інтервалом з'єднання, затримкою та таймаутом. Після успішного встановлення з'єднання обидва пристрої переходять до обміну даними відповідно до протоколу GATT. Для модуля HM-10 факт встановлення з'єднання індикується світлодіодом на платі – він перестає блимати і починає світитися постійно, а на виводі STATE встановлюється логічна одиниця.

Енергоефективність BLE

Низьке енергоспоживання BLE досягається завдяки комплексу архітектурних рішень. По-перше, пристрій більшу частину часу перебуває в режимі глибокого сну, прокидаючись лише на час передачі або прийому пакету даних. По-друге, тривалість одного пакету BLE є дуже малою – порядку кількох мілісекунд, після чого пристрій знову переходить у режим сну. По-третє, параметри з'єднання – інтервал з'єднання, затримка веденого пристрою (slave latency) та таймаут – можуть гнучко налаштовуватись для досягнення оптимального балансу між затримкою передачі даних та енергоспоживанням. Завдяки цим механізмам пристрій на базі BLE, що

передає дані раз на секунду, може споживати в середньому менше 10 мкА, що забезпечує роки автономної роботи від невеликої літієвої батареї типу CR2032.

Таким чином, протокол BLE 4.0 є добре продуманою і технічно зрілою технологією, що поєднує в собі ефективну модель організації даних через GATT, гнучкі механізми управління енергоспоживанням та широку підтримку з боку споживчих пристроїв. Ці властивості роблять BLE оптимальною основою для побудови систем бездротової телеметрії, де критично важливими є автономність, надійність зв'язку та можливість взаємодії з мобільними пристроями.

1.4 Порівняльний аналіз існуючих рішень на базі BLE-модулів

Сучасний ринок електронних компонентів пропонує широкий спектр готових модулів та мікросхем для побудови систем бездротового зв'язку на базі технології Bluetooth Low Energy. Вибір конкретного рішення суттєво впливає на характеристики кінцевого пристрою, складність розробки програмного забезпечення, вартість серійного виробництва та можливості масштабування системи. У даному підрозділі розглянуто найбільш поширені BLE-модулі та мікросхеми, що застосовуються для побудови систем телеметрії, проведено їх порівняльний аналіз та обґрунтовано вибір модуля на базі CC2541 як апаратної основи розроблюваної системи [14].

Модуль HM-10 (CC2541, Jinan Huamao Technology)

Модуль HM-10 є одним із найбільш широко застосовуваних BLE-модулів у сфері любительської та напівпрофесійної електроніки. Він побудований на базі мікроконтролера CC2541 виробництва Texas Instruments та містить у собі повний BLE-стек, записаний у внутрішню флеш-пам'ять мікроконтролера, рис.1.4. Взаємодія з модулем здійснюється через інтерфейс UART за допомогою набору AT-команд, що значно спрощує його

використання – розробнику не потрібно знати деталі реалізації BLE-протоколу на низькому рівні.

Рисунок 1.4 – Модуль HM-10

Модуль підтримує стандарт BLE 4.0, забезпечує дальність зв'язку до 100 метрів у відкритому просторі та споживає в активному режимі близько 9 мА. Напруга живлення модуля разом з перехідною платою складає від 3,0 до 5,0 В, що забезпечує сумісність як з 3,3-вольтовими, так і з 5-вольтовими мікроконтролерними платформами, зокрема з Arduino. Швидкість послідовного порту за замовчуванням становить 9600 бод і може бути змінена через AT-команду. Модуль підтримує режими ведучого та веденого пристрою, що дозволяє організовувати як з'єднання зі смартфоном, так і зв'язок між двома модулями.

До переваг HM-10 належать: низька вартість, простота підключення та програмування, широка документація та велика спільнота розробників, наявність перехідної плати з регулятором напруги. Основним обмеженням є те, що через UART-інтерфейс доступний лише один кастомний сервіс з однією характеристикою, що обмежує можливості реалізації складних протоколів обміну даними на рівні BLE [15].

Модуль HC-08 / HC-10 (CC2540/CC2541)

Модулі серії HC-08 та HC-10 є близькими аналогами HM-10 і також побудовані на базі мікроконтроллерів CC2540 та CC2541 відповідно. Вони мають схожий набір AT-команд та аналогічний принцип роботи через UART-інтерфейс, рис.1.5. Основна відмінність між ними полягає у версії підтримуваного стандарту та незначних відмінностях у наборі команд і параметрах за замовчуванням.

Рисунок 1.5 – Модуль HC-08

Модуль HC-08 на базі CC2540 підтримує стандарт BLE 4.0 та має схожі з HM-10 характеристики за дальністю та енергоспоживанням. На практиці модулі серії HC широко застосовуються як бюджетна альтернатива HM-10, однак якість виготовлення та стабільність прошивки у різних партій можуть суттєво відрізнятись, що є важливим фактором при виборі компонентів для відповідальних застосувань. Документація на ці модулі є менш повною порівняно з HM-10, що може ускладнити розробку [13].

Модуль nRF52832 (Nordic Semiconductor)

Мікросхема nRF52832 виробництва норвезької компанії Nordic Semiconductor є одним із найбільш технічно досконалих рішень для побудови BLE-пристроїв. Вона являє собою однокристальну систему (SoC), що поєднує 32-бітне процесорне ядро ARM Cortex-M4F з тактовою частотою 64 МГц, 512 кБ флеш-пам'яті, 64 кБ оперативної пам'яті та повнофункціональний

радіомодуль BLE 5.0 з підтримкою додаткових протоколів — ANT та 2,4 ГГц proprietary, рис. 1.6.

Рисунок 1.6 – Модуль nRF52832

На відміну від модулів на базі CC2541, де BLE-стек є прихованим від розробника, при роботі з nRF52832 розробник має повний доступ до всіх рівнів програмного стеку через офіційний SDK від Nordic Semiconductor. Це відкриває значно ширші можливості для кастомізації — можна реалізовувати довільну кількість сервісів та характеристик GATT, налаштовувати параметри з'єднання, реалізовувати складні алгоритми енергозбереження та використовувати всі периферійні модулі мікроконтролера. Мікросхема має вбудований 12-бітний АЦП, апаратні інтерфейси SPI, I2C, UART, PWM, що дозволяє підключати широкий спектр датчиків безпосередньо до неї без зовнішнього мікроконтролера [16].

Однак висока функціональність nRF52832 має і зворотній бік — суттєво вищий поріг входження для розробника. Налаштування та програмування цієї мікросхеми вимагає знання ARM-архітектури, досвіду роботи з SDK Nordic та розуміння архітектури BLE-стеку на низькому рівні. Вартість готових модулів на базі nRF52832 є вищою порівняно з HM-10, хоча і залишається доступною для більшості розробників. Це рішення є оптимальним для розробки комерційних продуктів з високими вимогами до функціональності та енергоефективності.

Модуль ESP32 (Espressif Systems)

Мікросхема ESP32 виробництва китайської компанії Espressif Systems є надзвичайно популярним рішенням у спільноті розробників завдяки поєднанню підтримки Wi-Fi та Bluetooth (включаючи BLE 4.2) в одному корпусі при відносно невисокій вартості. ESP32 містить двоядерний 32-бітний процесор Xtensa LX6 з тактовою частотою до 240 МГц, 520 кБ оперативної пам'яті, вбудований 12-бітний АЦП, підтримку численних периферійних інтерфейсів та широкий набір інструментів розробки, включаючи підтримку Arduino IDE, MicroPython та власного фреймворку ESP-IDF, рис. 1.7.

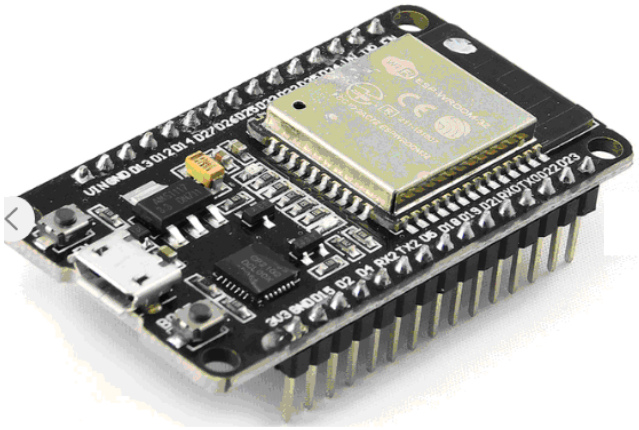


Рисунок 1.7 – Модуль ESP32

Завдяки підтримці Wi-Fi ESP32 може передавати дані як через BLE, так і безпосередньо до хмарних платформ через інтернет, що робить його привабливим вибором для IoT-застосувань. Однак це поєднання має свою ціну — енергоспоживання ESP32 є значно вищим порівняно з спеціалізованими BLE-мікросхемами, що робить його менш придатним для систем з суворими вимогами до автономності. Крім того, одночасна робота Wi-Fi та BLE може призводити до взаємних завад через використання одного радіодіапазону 2,4 ГГц.

Модуль RN4870/71 (Microchip Technology)

Модулі серії RN4870 та RN4871 виробництва компанії Microchip Technology є готовими сертифікованими BLE-модулями промислового класу, що підтримують стандарт BLE 4.2. Вони мають вбудований мікроконтролер, BLE-стек та управляються через UART за допомогою набору ASCII-команд, подібно до HM-10. Модулі сертифіковані за стандартами FCC, IC та CE, що спрощує процедуру сертифікації кінцевого пристрою.

Рисунок 1.8 – Модуль RN4870

Основна перевага модулів RN4870/71 — висока надійність, стабільність роботи та якість документації, що є важливим для промислових застосувань. Вони підтримують широкий набір стандартних GATT-профілів, включаючи UART Transparent Service, а також мають вбудований скриптовий рушій для реалізації простої логіки без зовнішнього мікроконтролера. Разом з тим, вартість цих модулів є суттєво вищою порівняно з бюджетними аналогами, що обмежує їх застосування в економічно чутливих проектах [17].

Порівняльний аналіз розглянутих рішень

Для систематизації результатів аналізу розглянутих BLE-модулів наведемо їх основні характеристики у порівняльній таблиці.

Таблиця 1.2 – Порівняльна характеристика BLE-модулів

Характеристика	**HM-10 (CC2541)**	**nRF52832**	**ESP32**	**RN4870**
Стандарт BLE	4.0	5.0	4.2	4.2
Інтерфейс керування	AT-команди	Власний SDK	Arduino/ESP-IDF	AT-команди
Енергоспоживання	Низьке	Дуже низьке	Середнє	Низьке
Поріг входження	Низький	Високий	Середній	Низький
Вартість модуля	Дуже низька	Середня	Низька	Висока
Потреба у зовн. МК	Так	Ні	Ні	Частково
Документація	Достатня	Відмінна	Відмінна	Відмінна
Застосування	DIY, навчання, прототипи	Комерційні продукти	IoT, прототипи	Промисловість

Обґрунтування вибору модуля CC2541 (HM-10)

За результатами проведеного порівняльного аналізу для реалізації системи телеметрії в рамках даної бакалаврської роботи обрано модуль HM-10 на базі мікроконтролера CC2541. Цей вибір обґрунтований сукупністю таких факторів.

По-перше, модуль HM-10 має надзвичайно низький поріг входження – управління ним здійснюється через прості AT-команди по UART, що не вимагає знання деталей реалізації BLE-стеку та дозволяє зосередитись на розробці логіки роботи системи телеметрії в цілому. По-друге, модуль забезпечує доступну сумісність зі смартфонами на базі Android та iOS, що дозволяє реалізувати мобільний інтерфейс моніторингу без додаткового обладнання. По-третє, низька вартість модуля робить розроблену систему економічно доступною та придатною для тиражування. По-четверте, наявність великої кількості документації, прикладів коду та активної спільноти розробників суттєво прискорює процес розробки та налагодження системи.

Таким чином, модуль HM-10 на базі CC2541 є оптимальним вибором для навчальних та дослідницьких проектів у галузі бездротової телеметрії, забезпечуючи достатній функціонал при мінімальній складності розробки та невисокій вартості апаратної реалізації.

1.5 Постановка задачі та вимоги до розроблюваної системи

Проведений у попередніх підрозділах аналіз предметної області, огляд технологій бездротової передачі даних та порівняльний аналіз існуючих BLE-модулів дозволяють сформулювати чітку постановку задачі для розроблюваної системи телеметрії та визначити комплекс вимог до її апаратної та програмної складових. Коректна постановка задачі є необхідною умовою для успішного проектування системи, оскільки саме вона визначає межі розробки, критерії оцінки результату та пріоритети при прийнятті проектних рішень.

Загальна постановка задачі

В рамках даної бакалаврської роботи необхідно розробити систему телеметрії, що забезпечує автоматизований збір даних від одного або кількох датчиків фізичних величин, їх первинну обробку на рівні мікроконтролерного вузлу та бездротову передачу до пристрою-приймача по каналу зв'язку на базі технології Bluetooth Low Energy з використанням модуля HM-10 на базі мікроконтролера CC2541. Система повинна забезпечувати можливість відображення телеметричних даних як на персональному комп'ютері, так і на мобільному пристрої під управлінням Android або iOS у режимі реального часу.

Розроблювана система має відповідати концепції мінімальної складності при максимальній функціональності — тобто забезпечувати виконання всіх необхідних функцій телеметрії з використанням доступної та недорогої елементної бази, з можливістю подальшого розширення та адаптації під

конкретні прикладні задачі. Система повинна бути відтворюваною, добре задокументованою та придатною для використання як у навчальних цілях, так і як основа для практичних застосувань у галузі автоматизації та робототехніки.

Функціональні вимоги до системи

Система повинна забезпечувати безперервний збір даних від підключених датчиків з налаштованою частотою опитування. Мінімальна частота опитування датчиків повинна становити не менше 1 Гц, що є достатнім для більшості задач моніторингу повільно змінюваних фізичних параметрів — температури, вологості, тиску. Для датчиків, що реєструють більш динамічні процеси, частота опитування повинна бути збільшена відповідно до вимог конкретного застосування.

Система повинна забезпечувати первинну обробку зібраних даних на рівні мікроконтролерного вузлу, що включає масштабування сирих значень АЦП до фізичних одиниць вимірювання, усереднення для зменшення впливу шумів та формування пакетів даних для передачі по BLE-каналу. Формат пакету даних повинен бути фіксованим та добре задокументованим, що забезпечить можливість подальшого розширення системи.

Система повинна забезпечувати надійну бездротову передачу телеметричних даних по каналу BLE на відстані не менше 10 метрів в умовах приміщення. Передача даних повинна здійснюватись у режимі реального часу з мінімальною затримкою від моменту вимірювання до відображення на пристрої-приймачі. У разі тимчасової втрати зв'язку система повинна автоматично відновлювати з'єднання без втручання оператора.

Система повинна забезпечувати відображення телеметричних даних на персональному комп'ютері у вигляді числових значень та графіків зміни параметрів у часі. Програма-клієнт для ПК повинна забезпечувати можливість збереження отриманих даних у файл для подальшого аналізу. Крім того, система повинна забезпечувати можливість перегляду поточних значень

телеметричних параметрів на мобільному пристрої з використанням стандартних BLE-додатків або спеціалізованого мобільного застосунку.

Технічні вимоги до системи

Технічні вимоги визначають кількісні характеристики системи та обмеження на її реалізацію. На основі аналізу цільового призначення системи та характеристик обраної елементної бази сформульовано такі технічні вимоги.

Напруга живлення мікроконтролерного вузлу повинна становити 5 В від USB-джерела або від автономного акумулятора з відповідним стабілізатором напруги. Споживаний струм всього вузлу в активному режимі роботи не повинен перевищувати 100 мА, що забезпечить можливість тривалої автономної роботи від стандартного портативного зарядного пристрою ємністю 2000 мАг протягом не менше 20 годин.

Дальність бездротового зв'язку між передавальним вузлом та пристроєм-приймачем повинна становити не менше 10 метрів в умовах приміщення з типовими перешкодами у вигляді меблів та перегородок. У відкритому просторі без перешкод дальність зв'язку повинна забезпечуватись на відстані не менше 30 метрів. Ці вимоги відповідають типовим умовам застосування системи телеметрії у лабораторних та виробничих приміщеннях.

Час встановлення BLE-з'єднання від моменту ввімкнення передавального вузлу до початку прийому даних на пристрої-приймачі не повинен перевищувати 10 секунд. Затримка передачі даних від моменту вимірювання до відображення на екрані приймача не повинна перевищувати 1 секунду при частоті оновлення даних 1 Гц. Стабільність з'єднання повинна забезпечувати безперервну роботу без втрат пакетів протягом не менше 1 години в умовах типового радіоефіру приміщення.

Вимоги до апаратної частини

Апаратна частина системи телеметрії повинна будуватись на базі доступних та широко розповсюджених компонентів, що забезпечить

можливість відтворення системи без залучення спеціалізованого обладнання. Мікроконтролерний вузол повинен базуватись на платформі Arduino або сумісній платі з мікроконтролером AVR або ARM, що забезпечить достатню обчислювальну потужність для виконання всіх функцій збору та обробки даних.

Модуль бездротового зв'язку повинен бути побудований на базі мікроконтролера CC2541 та підключатись до мікроконтролерного вузлу через інтерфейс UART. Підключення датчиків повинно здійснюватись через стандартні інтерфейси — аналоговий вхід АЦП, I2C або SPI — залежно від типу датчика. Принципова електрична схема системи повинна бути розроблена з дотриманням вимог електромагнітної сумісності та забезпечувати захист входів мікроконтролера від перевищення допустимої напруги.

Вимоги до програмного забезпечення

Програмне забезпечення системи телеметрії повинно відповідати принципам модульності та розширюваності. Код мікроконтролерного вузлу повинен бути структурований таким чином, щоб додавання нових типів датчиків або зміна формату пакету даних вимагали мінімальних змін у програмі. Програма-клієнт повинна мати зрозумілий та інтуїтивно зрозумілий інтерфейс користувача, що не вимагає спеціальної підготовки для роботи з системою.

Весь програмний код повинен бути детально прокоментований та супроводжуватись технічною документацією, що описує принципи роботи кожного модуля, формат пакетів даних та процедуру налаштування системи. Це забезпечить можливість подальшої підтримки та розвитку системи іншими розробниками.

Структура розроблюваної системи

На основі сформульованих вимог визначено загальну структуру розроблюваної системи телеметрії, що складається з трьох основних

підсистем. Перша підсистема — вимірювальний вузол, що включає мікроконтролерну плату, датчики фізичних величин та модуль НМ-10 як засіб бездротової передачі даних. Друга підсистема — канал бездротового зв'язку на базі BLE 4.0, що забезпечує передачу телеметричних пакетів від вимірювального вузлу до пристрою-приймача. Третя підсистема — програмне забезпечення приймальної сторони, що реалізує прийом, декодування, відображення та збереження телеметричних даних на персональному комп'ютері або мобільному пристрої.

Таким чином, постановка задачі та сформульований комплекс вимог визначають чіткі межі розробки та критерії оцінки результату. Виконання всіх сформульованих вимог забезпечить створення функціональної, надійної та практично придатної системи телеметрії на базі Bluetooth-модуля CC2541, що відповідає сучасним вимогам до систем бездротового моніторингу в галузі автоматизації та робототехніки.

2 РОЗРОБЛЕННЯ АПАРАТНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ТЕЛЕМЕТРІЇ

2.1 Мікроконтролер CC2541: архітектура, технічні характеристики

Мікроконтролер CC2541 виробництва компанії Texas Instruments є спеціалізованою однокристальною системою (System-on-Chip, SoC), розробленою для застосувань бездротового зв'язку стандарту Bluetooth Low Energy 4.0. Він поєднує в єдиному корпусі повнофункціональний мікроконтролер на базі архітектури 8051, радіочастотний приймач-передавач діапазону 2,4 ГГц та розвинуту периферію, що робить його самодостатнім рішенням для побудови автономних бездротових пристроїв без застосування додаткових зовнішніх мікросхем. Завдяки оптимальному поєднанню функціональності, низького енергоспоживання та доступної вартості CC2541 набув широкого застосування у системах телеметрії, медичних пристроях, портативній електроніці та промисловому інтернеті речей.

Архітектура процесорного ядра

В основі CC2541 лежить вдосконалене ядро мікроконтролера архітектури 8051, що функціонує на тактовій частоті до 32 МГц. Архітектура 8051 є однією з найбільш перевірених і широко застосовуваних у вбудованих системах — вона відрізняється чіткою організацією, детально документована та має розвинуту екосистему інструментів розробки. У реалізації Texas Instruments класичне ядро 8051 було суттєво вдосконалено: збільшено продуктивність за рахунок скорочення кількості тактів на виконання більшості інструкцій порівняно зі стандартним ядром 8051, розширено адресний простір пам'яті та додано підтримку DMA-передач для ефективної роботи з

периферійними модулями. Загальна структура мікроконтролера CC2541 наведена на рисунку 2.1.

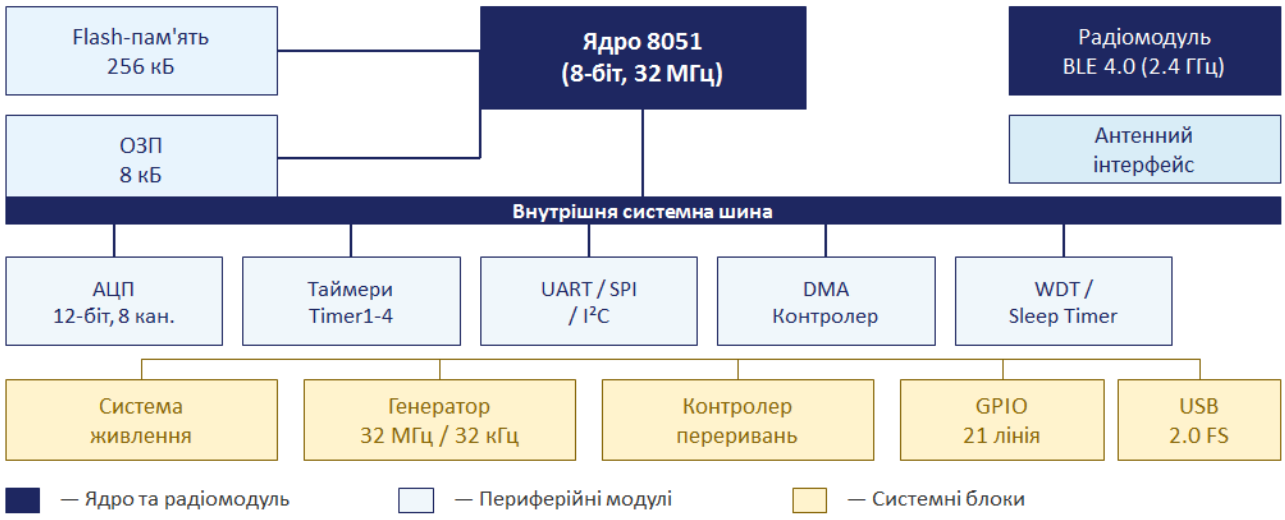


Рисунок 2.1 – Структурна схема мікроконтролера CC2541

Як видно зі структурної схеми, всі функціональні блоки мікроконтролера об'єднані внутрішньою системною шиною, через яку здійснюється обмін даними між процесорним ядром, пам'яттю, периферійними модулями та радіомодулем. Такий підхід забезпечує ефективну взаємодію всіх блоків при мінімальному часі доступу та дозволяє реалізовувати DMA-передачі між периферією та пам'яттю без участі процесорного ядра, що суттєво підвищує загальну продуктивність системи та знижує енергоспоживання [14,15].

Організація пам'яті

CC2541 має розвинуту систему пам'яті, що включає два основних типи: Flash-пам'ять для зберігання програмного коду та констант і оперативну пам'ять (SRAM) для зберігання змінних та стеку під час виконання програми. Детальна карта розподілу пам'яті мікроконтролера наведена на рисунку 2.2.

Обсяг Flash-пам'яті складає 256 кБ. Адресний простір Flash організований таким чином, що молодші адреси (0x0000–0x07FF) зарезервовані під таблицю векторів переривань, основний діапазон (0x0800–

0xEFFF) використовується для зберігання коду користувацької програми, а старші адреси відведені під BLE-стек (0xF000–0xF7FF) та системні регістри (0xF800–0xFFFF). Оскільки адресний простір процесора 8051 обмежений 64 кБ, для адресації всього обсягу Flash у CC2541 реалізовано механізм банківської комутації (bank switching), що дозволяє відображати різні банки Flash у єдиний адресний простір процесора.

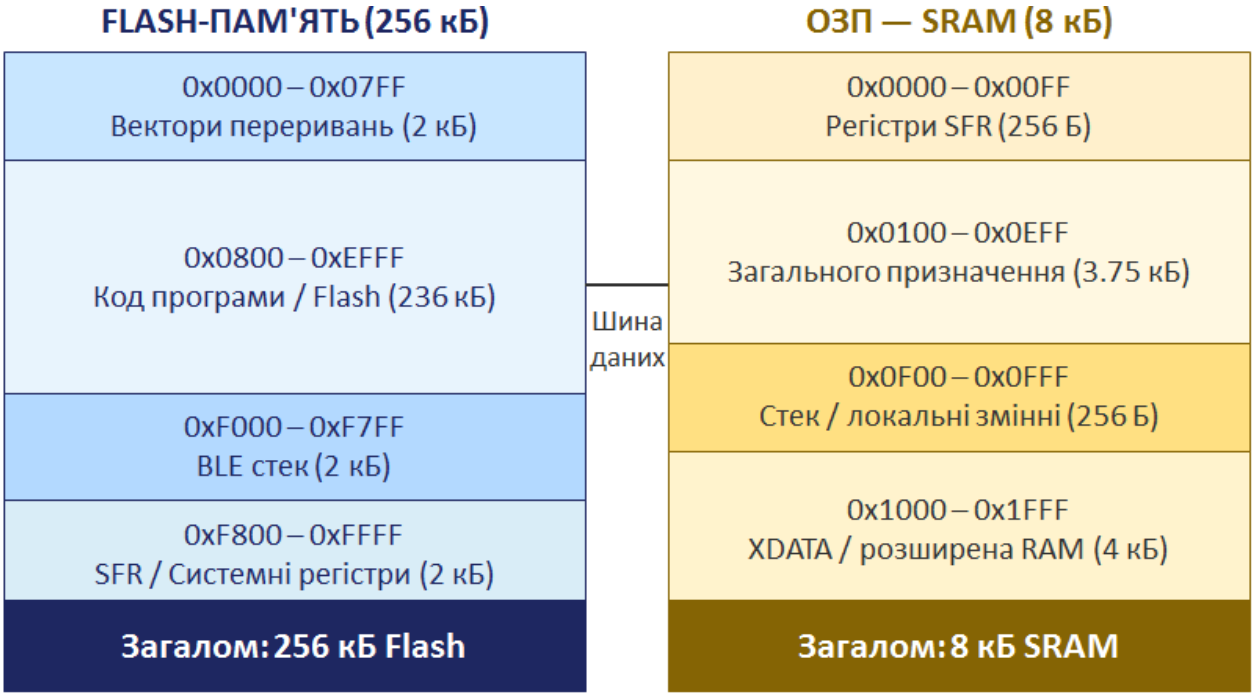


Рисунок 2.2 – Карта розподілу пам'яті мікроконтролера CC2541

Обсяг оперативної пам'яті SRAM складає 8 кБ. Вона поділяється на кілька функціональних областей: регістри спеціального призначення SFR (256 байт), область пам'яті загального призначення (3,75 кБ), область стеку та локальних змінних (256 байт) та розширена XDATA-пам'ять (4 кБ), доступна через відповідні інструкції 8051. Окрім цього, CC2541 має 2 кБ пам'яті IEEE MAC-адреси та унікального ідентифікатора пристрою, що зберігаються у виділеній захищеній ділянці і не можуть бути змінені користувачем.

Радіочастотний модуль

Вбудований радіомодуль CC2541 є повнофункціональним приймачем-передавачем стандарту Bluetooth Low Energy 4.0, що працює в

неліцензованому діапазоні частот ISM 2,400–2,4835 ГГц. Радіомодуль підтримує всі 40 каналів BLE, включаючи три рекламних канали (37, 38, 39) та 37 каналів для передачі даних, а також реалізує механізм адаптивного стрибкоподібного перестроювання частоти (FHSS) для підвищення завадостійкості зв'язку.

Для модуляції сигналу застосовується метод GFSK (Gaussian Frequency Shift Keying) зі швидкістю передачі даних 1 Мбіт/с на фізичному рівні. Чутливість приймача складає -93 дБм при BER (Bit Error Rate) 0,1%, що забезпечує надійний зв'язок на відстані до 100 метрів у відкритому просторі при вихідній потужності передавача 0 дБм. Вихідна потужність передавача програмно налаштовується в діапазоні від -23 дБм до +6 дБм, що дозволяє гнучко регулювати компроміс між дальністю зв'язку та рівнем енергоспоживання.

Радіомодуль підключається до зовнішньої антени або антенного з'єднувача через вбудований балун (балансно-небалансний перетворювач) та фільтр. Мікросхема CC2541 підтримує диференціальне антенне підключення, що покращує завадостійкість та знижує вплив паразитних випромінювань на роботу пристрою.

Периферійні модулі

CC2541 оснащений розвинутою периферією, що забезпечує можливість підключення широкого спектру зовнішніх компонентів та датчиків без застосування додаткових мікросхем. Аналого-цифровий перетворювач (АЦП) має роздільну здатність 12 біт та підтримує до 8 аналогових вхідних каналів. АЦП може працювати як у режимі одиночного перетворення, так і у режимі автоматичного сканування кількох каналів з програмно налаштовуваною частотою вибірки. Наявність вбудованого АЦП є ключовою перевагою CC2541 для застосувань телеметрії, оскільки дозволяє безпосередньо підключати аналогові датчики температури, тиску, освітленості та інших фізичних величин.

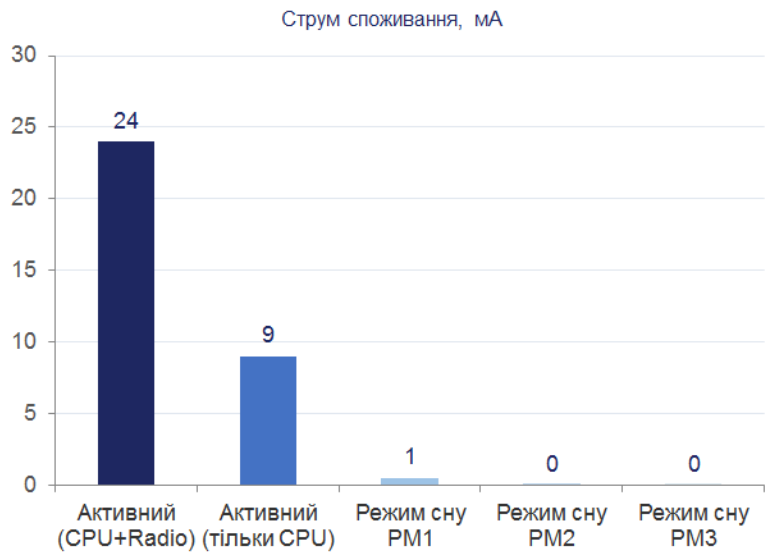
Для організації зв'язку з зовнішніми пристроями CC2541 має два програмовані послідовних порти USART, кожен з яких може бути налаштований як UART або SPI залежно від вимог застосування. Апаратна підтримка I2C дозволяє підключати цифрові датчики та мікросхеми пам'яті з цим інтерфейсом. Наявність двох незалежних послідовних портів є суттєвою перевагою для систем телеметрії, де один порт використовується для зв'язку з датчиками, а інший – для керування модулем HM-10.

Таймерна підсистема CC2541 включає чотири таймери різного призначення: Timer 1 є 16-бітним таймером з підтримкою режимів ШІМ та захоплення/порівняння, Timer 2 є спеціалізованим таймером для синхронізації роботи BLE-протоколу (MAC Timer), Timer 3 та Timer 4 є 8-бітними загального призначення. Окрім цього, є таймер сну (Sleep Timer), що функціонує навіть у режимах глибокого сну і використовується для пробудження мікроконтролера у заданий час.

Кількість ліній загального призначення GPIO складає 21, що є достатнім для підключення більшості поширених датчиків та виконавчих пристроїв. Всі лінії GPIO мають програмно налаштовувані внутрішні підтягуючі резистори та підтримку переривань по фронту або спаду сигналу. Контролер прямого доступу до пам'яті (DMA) має 5 незалежних каналів і дозволяє виконувати передачу даних між периферією та пам'яттю без участі процесора, що є важливим для підвищення енергоефективності при роботі з великими масивами даних від АЦП.

Режими енергоспоживання

Одним із ключових параметрів CC2541 є підтримка розвинутої системи управління живленням, що включає кілька режимів з різним рівнем енергоспоживання. Порівняльна характеристика всіх режимів наведена на рисунку 2.3.



Режим	Струм	Опис
Активний (CPU+Radio)	~24 мА	Повна робота: ядро + BLE-радіо активні
Активний (CPU)	~9 мА	Ядро працює, радіо вимкнено
PM1 (сон)	~0.5 мкА	CPU зупинено, таймери активні
PM2 (глибокий сон)	~0.003 мкА	Тільки RTC та RAM живляться
PM3 (вимкнено)	~0.0003 мкА	Лише зовнішнє переривання

Рисунок 2.3 - Режими енергоспоживання мікроконтролера CC2541

У активному режимі при одночасній роботі процесорного ядра та радіомодуля загальний струм споживання становить близько 24 мА. При вимкненому радіомодулі та активному ядрі струм знижується до 9 мА. Режим PM1 забезпечує збереження вмісту регістрів та оперативної пам'яті при зупиненому процесорному ядрі, а струм споживання складає близько 0,5 мкА. Режим PM2 є більш глибоким сном – зберігається лише вміст оперативної пам'яті та продовжує працювати таймер реального часу (RTC); струм споживання знижується до 0,003 мкА. Найглибший режим сну PM3 зберігає лише вміст регістрів спеціального призначення; пробудження можливе лише за зовнішнім перериванням, а струм споживання складає близько 0,0003 мкА, що відповідає рівню менше 1 мкА.

Такий широкий діапазон режимів енергоспоживання дозволяє гнучко оптимізувати споживання системи телеметрії залежно від вимог до частоти вимірювань та дальності зв'язку. Наприклад, при частоті передачі телеметричних даних 1 раз на секунду пристрій на базі CC2541 більшу частину часу може перебувати в режимі PM2, прокидаючись лише на час вимірювання та передачі пакету, що дозволяє досягти середнього струму

споживання на рівні одиниць мікроампер і забезпечити роки автономної роботи від невеликої батареї.

Технічні характеристики CC2541

Для зручності подальшого використання у проектуванні системи телеметрії зведемо основні технічні характеристики мікроконтролера CC2541 до таблиці 2.1.

Таблиця 2.1 – Основні технічні характеристики мікроконтролера CC2541

Параметр	**Значення**
Процесорне ядро	8051 (вдосконалене), 8-біт
Тактова частота	до 32 МГц
Flash-пам'ять	256 кБ
Оперативна пам'ять (SRAM)	8 кБ
Стандарт бездротового зв'язку	Bluetooth Low Energy 4.0
Радіодіапазон	2,400 – 2,4835 ГГц
Швидкість передачі даних	1 Мбіт/с
Чутливість приймача	-93 дБм
Вихідна потужність передавача	-23 … +6 дБм
АЦП	12-біт, 8 каналів
Послідовні інтерфейси	2 × UART/SPI, I2C
GPIO	21 лінія
Таймери	Timer1 (16-біт), Timer2-4, Sleep Timer
DMA	5 каналів
Напруга живлення	2,0 – 3,6 В
Корпус	QFN-40 (6×6 мм)
Діапазон робочих температур	-40 … +85 °C

Таким чином, мікроконтролер CC2541 є повнофункціональним SoC-рішенням для побудови пристроїв BLE, що поєднує в собі все необхідне для

реалізації системи телеметрії – процесорне ядро, пам'ять, аналогову та цифрову периферію, а також повнофункціональний радіомодуль BLE 4.0. Ці властивості роблять CC2541 оптимальним вибором як апаратну основу розроблюваної системи телеметрії в рамках даної бакалаврської роботи.

2.2 Модуль HM-10 на базі CC2541

Модуль HM-10 є готовим апаратним рішенням для організації бездротового зв'язку за стандартом Bluetooth Low Energy 4.0, побудованим на базі мікроконтролера CC2541 виробництва Texas Instruments. Модуль розроблений китайською компанією Jinan Huamao Technology і є одним із найбільш поширених BLE-модулів у сфері вбудованих систем, робототехніки та систем телеметрії завдяки поєднанню низької вартості, простоти використання та достатньої функціональності для широкого кола практичних застосувань. Ключова концепція модуля полягає в тому, що весь BLE-стек реалізований на рівні вбудованого програмного забезпечення мікроконтролера CC2541, а взаємодія з зовнішнім мікроконтролером або комп'ютером здійснюється через простий і добре задокументований інтерфейс AT-команд по UART. Це дозволяє розробнику використовувати модуль як «чорний ящик», не заглиблюючись у деталі реалізації BLE-протоколу [14,15].

Будова та конструктивне виконання модуля

Конструктивно модуль HM-10 являє собою невелику друковану плату розмірами приблизно 27×13 мм, на якій розміщено мікросхему CC2541F256, кварцовий резонатор, антену (як правило, друковану або чіпову), пасивні компоненти та світлодіод індикації стану з'єднання. Модуль, як правило, постачається у вигляді перехідної плати з регулятором напруги та буферними елементами на лініях UART, що дозволяє безпосередньо підключати його до плат Arduino та інших мікроконтролерних систем з рівнями напруги 5 В без

ризику пошкодження мікросхеми CC2541, яка розрахована на напругу живлення 3,3 В, рис. 2.4.

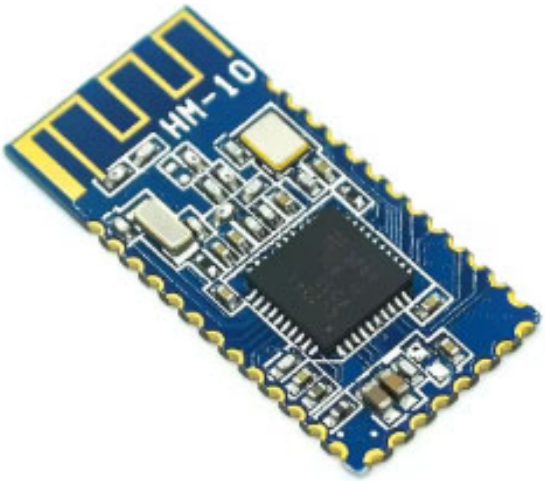


Рисунок 2.4 – Модуль CC2541

Вбудований у прошивку CC2541 BLE-стек реалізує всі необхідні рівні протоколу — від фізичного рівня до рівня GATT, надаючи назовні один кастомний сервіс з характеристиками читання та запису, що функціонує як прозорий послідовний канал передачі даних. Такий підхід суттєво спрощує розробку, оскільки дозволяє передавати довільні байтові послідовності через BLE-з'єднання аналогічно до звичайного послідовного порту.

Розпіновка модуля

Модуль HM-10 має шість основних виводів, розташування та призначення яких наведено на рисунку 2.5.

Вивід BRK/EN призначений для примусового розриву активного BLE-з'єднання шляхом подачі низького логічного рівня. Слід зазначити, що ця функція реалізована не у всіх версіях прошивки модуля, тому перед використанням рекомендується перевірити версію прошивки командою AT+VERS?. Вивід VCC є входом живлення модуля і підключається до джерела постійної напруги в діапазоні від 3,0 до 5,0 В. Вивід GND є спільною

шиною – загальним проводом, який повинен бути з'єднаний із загальним проводом керуючого мікроконтролера та джерела живлення.

Рисунок 2.5 – Розпіновка модуля HM-10 на базі CC2541

Виводи TX та RX реалізують інтерфейс UART для обміну даними з керуючим мікроконтролером або комп'ютером. Важливо враховувати, що вивід TX модуля підключається до виводу RX керуючої плати, і навпаки – вивід RX модуля підключається до виводу TX керуючої плати, що є стандартним правилом з'єднання UART-пристроїв. Швидкість передачі за замовчуванням складає 9600 бод, однак може бути змінена командою AT+BAUD у діапазоні від 1200 до 230400 бод. Вивід STATE відображає поточний стан BLE-з'єднання: при відсутності з'єднання на ньому формується меандр з частотою 1 Гц (що дублює мигання світлодіода), а при встановленому з'єднанні – постійний високий логічний рівень.

Технічні характеристики модуля

Модуль HM-10 має такі основні технічні характеристики. Напруга живлення модуля з перехідною платою складає від 3,0 до 5,0 В, максимальний споживаний струм – 50 мА. У активному режимі роботи з встановленим BLE-з'єднанням струм споживання становить близько 9 мА, а в режимі сну – від 50 до 200 мкА, що робить модуль придатним для автономних застосувань з

батарейним живленням. Вихідна потужність радіопередавача може бути налаштована на одне з чотирьох значень: -23 дБм, -6 дБм, 0 дБм або +6 дБм, що відповідає дальності зв'язку від кількох метрів до 100 метрів у відкритому просторі.

Схема підключення до мікроконтролерної плати

Для підключення модуля HM-10 до мікроконтролерної плати Arduino UNO необхідно з'єднати відповідні виводи згідно зі схемою, наведеною на рисунку 2.6.

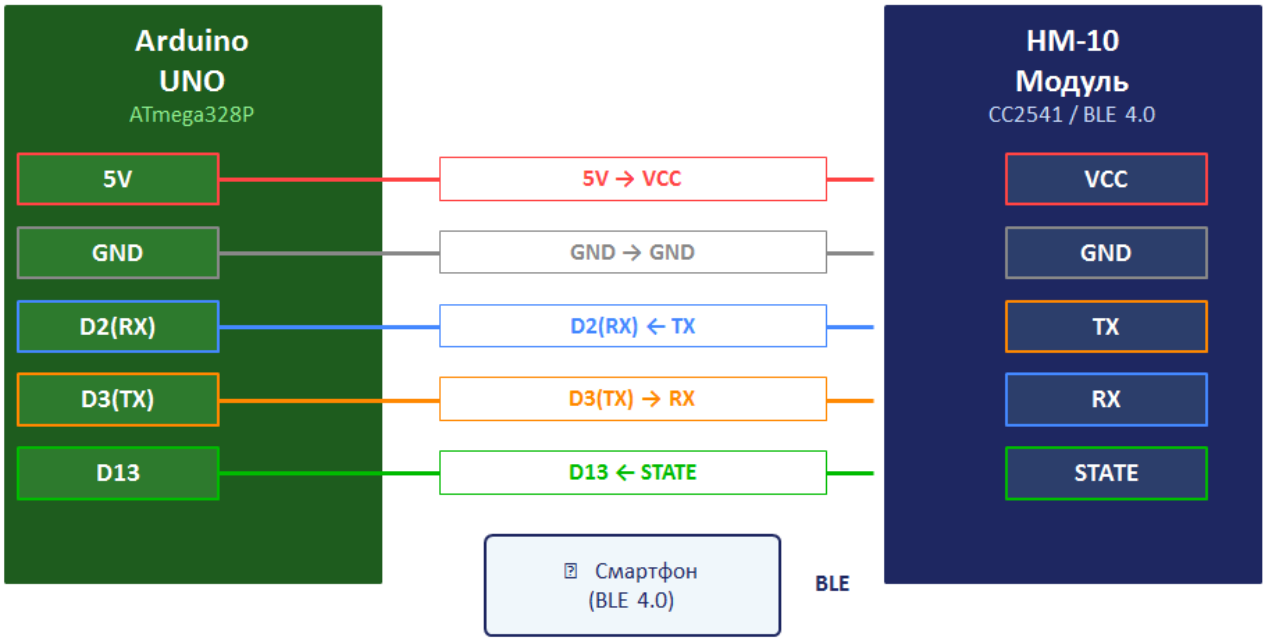


Рисунок 2.6 - Схема підключення модуля HM-10 до мікроконтролерної плати

Як видно зі схеми, вивід VCC модуля підключається до виходу 5 В плати Arduino, вивід GND – до загального проводу. Виводи TX та RX модуля підключаються до програмних послідовних портів Arduino – відповідно до виводів D2 (RX) та D3 (TX). Використання бібліотеки SoftwareSerial дозволяє реалізувати програмний послідовний порт на будь-яких цифрових виводах Arduino, звільняючи апаратний UART (виводи D0/D1) для відлагодження через монітор порту. Вивід STATE підключається до виводу D13 для моніторингу стану з'єднання в програмі.

При такому підключенні модуль HM-10 виступає в ролі прозорого бездротового каналу між мікроконтролером Arduino та зовнішнім пристроєм – смартфоном або іншим модулем HM-10. Дані, надіслані мікроконтролером по UART до модуля, автоматично передаються через BLE-з'єднання до підключеного пристрою, і навпаки.

Варіанти топологій підключення

Модуль HM-10 підтримує кілька варіантів топологій бездротового з'єднання, що суттєво розширює можливості його застосування в системах телеметрії. Усі три основні топології наведені на рисунку 2.7.

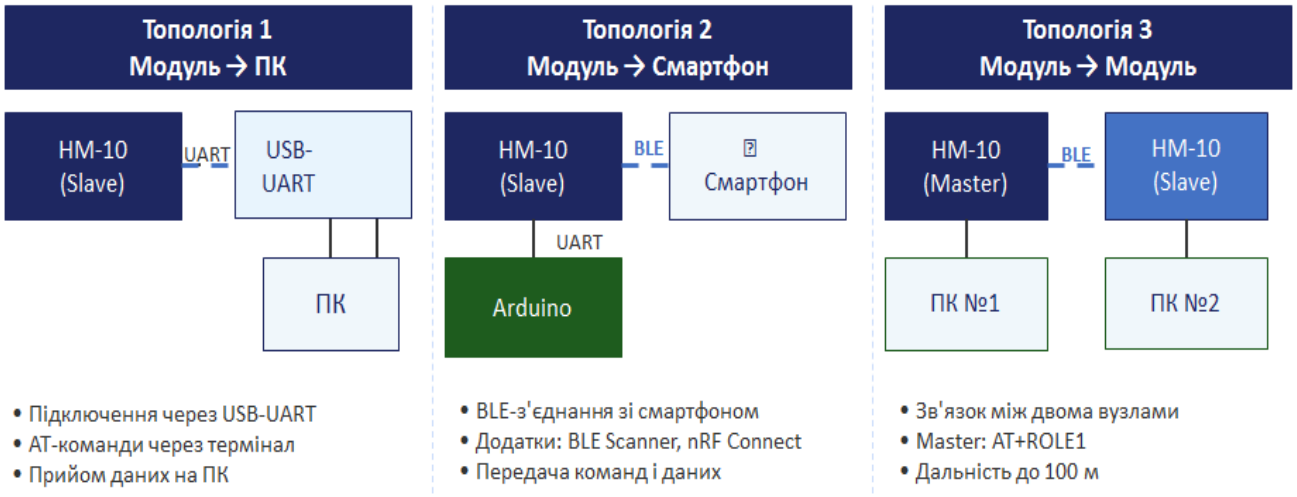


Рисунок 2.7 – Варіанти топологій підключення модуля HM-10

У першій топології модуль підключається до персонального комп'ютера через перетворювач USB-UART на базі мікросхеми FT232 або аналогічної. Це дозволяє керувати модулем безпосередньо з термінальної програми на ПК, надсилати AT-команди та отримувати дані через BLE-канал від підключеного смартфона або іншого модуля.

У другій топології модуль HM-10 підключається до мікроконтролерної плати Arduino і забезпечує бездротовий зв'язок зі смартфоном. Смартфон підключається до модуля за допомогою стандартних BLE-додатків – BLE Scanner, nRF Connect або LightBlue – і може як отримувати телеметричні дані

від Arduino, так і надсилати команди керування. Ця топологія є основною для розроблюваної системи телеметрії.

У третій топології два модулі HM-10 утворюють пряме бездротове з'єднання між собою, де один виступає в ролі ведучого (Master/Central), а інший – веденого (Slave/Peripheral). Для цього один з модулів перемикається в режим Master командою AT+ROLE1. Після встановлення з'єднання дані, надіслані через UART до будь-якого з модулів, прозоро передаються до іншого модуля через BLE-канал, що дозволяє реалізувати бездротову заміну провідного послідовного з'єднання між двома пристроями.

2.3 Інтерфейс UART та система AT-команд для модуля HM-10

Взаємодія між керуючим мікроконтролером та модулем HM-10 здійснюється через інтерфейс UART (Universal Asynchronous Receiver-Transmitter) — один із найбільш поширених стандартів послідовної передачі даних у вбудованих системах. UART є асинхронним інтерфейсом, що не потребує окремої тактової лінії: синхронізація між передавачем та приймачем забезпечується завдяки заздалегідь погодженій швидкості передачі та стандартній структурі кадру даних. Простота, надійність та мінімальна кількість сигнальних ліній (лише TX та RX) роблять UART ідеальним вибором для організації зв'язку між мікроконтролером і периферійним модулем [15].

Параметри UART-інтерфейсу модуля HM-10

Модуль HM-10 використовує UART з такими параметрами за замовчуванням: швидкість передачі 9600 бод, формат кадру 8N1 (8 біт даних, без біту паритету, 1 стоп-біт). Рівні напруги на виводах TX та RX мікросхеми CC2541 відповідають стандарту TTL 3,3 В, однак перехідна плата модуля містить рівнеперетворювач, що забезпечує сумісність з 5-вольтовими платформами, зокрема Arduino UNO на базі ATmega328P. Завдяки цьому

виводи TX та RX модуля можна безпосередньо підключати до відповідних виводів Arduino без додаткових елементів.

Формат UART-кадру та часова діаграма обміну між Arduino та модулем HM-10 при передачі AT-команди наведені на рисунку 2.7. Кожний байт даних передається у вигляді послідовності: стартовий біт (низький рівень), 8 біт даних (молодший біт першим, LSB first), стоп-біт (високий рівень). При швидкості 9600 бод тривалість одного біту становить приблизно 104 мкс, а тривалість передачі одного байту — близько 1,04 мс.

А. Формати AT-команд

Тип команди	Команда	Відповідь
Тест зв'язку	`AT`	`OK`
Запит параметра	`AT+<CMD>?`	`OK+<CMD>: <value>`
Встановлення	`AT+<CMD> <value>`	`OK+Set: <value>`
Виконання дії	`AT+<CMD>`	`OK+<CMD>`

Б. Часова діаграма обміну UART (9600, 8N1)

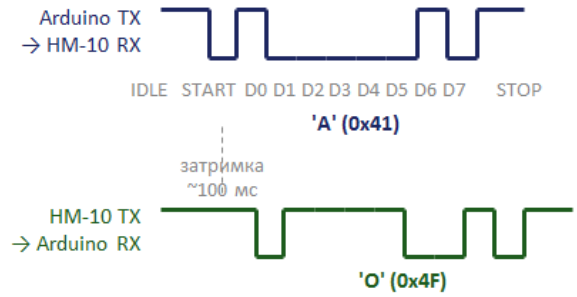


Рисунок 2.8 - Формат AT-команд модуля HM-10 та діаграма обміну по UART

Важливою особливістю роботи модуля HM-10 є те, що він має два принципово різних режими роботи по UART: режим команд та режим даних. У режимі команд модуль не має активного BLE-з'єднання і очікує надходження AT-команд для налаштування або керування. У режимі даних модуль має активне BLE-з'єднання і прозоро передає всі байти, що надходять по UART, через BLE-канал до підключеного пристрою, і навпаки. Перемикання між режимами відбувається автоматично при встановленні або розриві BLE-з'єднання [23].

Формат AT-команд

AT-команди (від англ. ATtention) – це текстовий протокол керування, що бере свій початок від команд модемів стандарту Hayes і широко застосовується в різноманітних комунікаційних модулях, зокрема GSM, Wi-Fi та Bluetooth. Усі команди починаються з префіксу «AT» і можуть мати різний

синтаксис залежно від призначення. Існує чотири основних формати команд, як наведено на рисунку 2.8.

Перший формат – тест зв'язку: команда «AT» без жодних параметрів надсилається для перевірки наявності зв'язку з модулем. У відповідь модуль повертає рядок «OK». Цю команду рекомендується надсилати на початку роботи для переконання у правильності підключення та налаштування UART.

Другий формат – запит поточного значення параметра: команда виду «AT+<CMD>?», де <CMD> - назва параметра. Модуль повертає відповідь «OK+<CMD>:<value>», де <value> - поточне значення параметра. Наприклад, команда «AT+BAUD?» повертає «OK+BAUD:0», що означає поточну швидкість 9600 бод.

Третій формат – встановлення нового значення параметра: команда виду «AT+<CMD><value>» без роздільника між назвою та значенням. Модуль підтверджує встановлення відповіддю «OK+Set:<value>». Наприклад, команда «AT+BAUD4» встановлює швидкість 115200 бод.

Четвертий формат – виконання дії: команда виду «AT+<CMD>» без значення, що запускає певну дію, наприклад скидання налаштувань (AT+RENEW) або перезапуск модуля (AT+RESET).

Принципово важливою особливістю AT-команд модуля HM-10 є те, що вони надсилаються без символу завершення рядка (\r\n), що відрізняє їх від команд деяких інших модулів. Модуль розпізнає команди за часовою затримкою: якщо після останнього прийнятого байту пройшло більше ~100 мс і рядок починається з «AT», він інтерпретується як команда. Час відповіді модуля на команду, як правило, становить від 50 до 200 мс.

Повний перелік основних AT-команд

Усі AT-команди модуля HM-10 можна поділити на чотири функціональні групи, що наведені на рисунку 2.9.

Група 1 — Загальні команди		
Команда	**Відповідь модуля**	**Опис**
AT	OK	Перевірка зв'язку з модулем
AT+VERS?	OK+VERS:HMSoft V540	Версія прошивки
AT+ADDR?	OK+ADDR:xx:xx:xx:xx:xx:xx	MAC-адреса модуля
AT+RESET	OK+RESET	Перезапуск модуля
AT+RENEW	OK+RENEW	Відновлення заводських налаштувань

Група 2 — Налаштування UART		
Команда	**Відповідь модуля**	**Опис**
AT+BAUD?	OK+BAUD:0	Запит поточної швидкості
AT+BAUD0	OK+Set:0	9600 бод
AT+BAUD1	OK+Set:1	19200 бод
AT+BAUD2	OK+Set:2	38400 бод
AT+BAUD4	OK+Set:4	115200 бод

Група 3 — Управління з'єднанням та режимом		
Команда	**Відповідь**	**Опис**
AT+ROLE?	OK+ROLE:0	Запит поточної ролі
AT+ROLE0	OK+Set:0	Режим Slave (веденого)
AT+ROLE1	OK+Set:1	Режим Master (ведучого)
AT+DISC?	OK+DISC:...	Сканування BLE-пристроїв
AT+CONxxxxxxxx	OK+CONNA	З'єднання з пристроєм (Master)
AT+IMME0	OK+Set:0	Авто-підключення при старті
AT+IMME1	OK+Set:1	Очікування команди AT+START

Група 4 — Налаштування BLE		
Команда	**Відповідь**	**Опис**
AT+NAME?	OK+NAME:HMSoft	Запит імені пристрою
AT+NAMEmydev	OK+Set:mydev	Встановлення імені
AT+PIN?	OK+PIN:000000	Запит PIN-коду
AT+NOTI1	OK+Set:1	Сповіщення про з'єднання
AT+POWE?	OK+POWE:2	Запит потужності TX
AT+POWE2	OK+Set:2	Встановлення потужності (0 дБм)

Рисунок 2.9 - Основні AT-команди модуля HM-10

До першої групи – загальних команд – належать команди перевірки зв'язку, отримання інформації про версію прошивки та MAC-адресу модуля, а також команди перезапуску та відновлення заводських налаштувань. Команда AT+ADDR? є особливо важливою при побудові систем з кількома модулями, оскільки дозволяє отримати унікальну MAC-адресу конкретного модуля для подальшого використання в команді AT+CON для адресного підключення в режимі Master.

До другої групи – команд налаштування UART – належать команди зміни швидкості передачі. Підтримувані швидкості: 1200 (0), 2400 (1), 4800 (2), 9600 (3), 19200 (4), 38400 (5), 57600 (6), 115200 (7), 230400 (8). Після зміни швидкості командою AT+BAUDx необхідно перезапустити модуль командою AT+RESET, після чого UART буде працювати на новій швидкості. Важливо одночасно змінити швидкість і в програмі керуючого мікроконтролера, інакше зв'язок буде втрачено.

До третьої групи – команд управління з'єднанням та режимом – належать команди перемикання ролі модуля між Master та Slave, сканування доступних BLE-пристроїв та встановлення з'єднання за MAC-адресою. Команда AT+DISC? в режимі Master повертає список знайдених BLE-пристроїв у форматі «OK+DISC:0:MAC1», «OK+DISC:1:MAC2» і т.д. Після отримання MAC-адреси потрібного пристрою з'єднання встановлюється командою AT+CON<MAC>, де <MAC> — шестибайтова MAC-адреса без роздільників.

До четвертої групи – команд налаштування BLE – належать команди зміни імені пристрою, PIN-коду, налаштування сповіщень про стан з'єднання та вихідної потужності передавача. Команда AT+NOTI1 вмикає режим сповіщень, при якому модуль надсилає по UART рядки «OK+CONN» при встановленні з'єднання та «OK+LOST» при його розриві – це дозволяє програмі мікроконтролера реагувати на зміну стану з'єднання без необхідності постійного опитування виводу STATE [23].

Алгоритм ініціалізації модуля

Для коректної роботи системи телеметрії необхідно виконати початкове налаштування модуля HM-10 згідно з алгоритмом, наведеним на рисунку 2.10. Алгоритм передбачає послідовне виконання кількох кроків після подачі живлення на модуль.

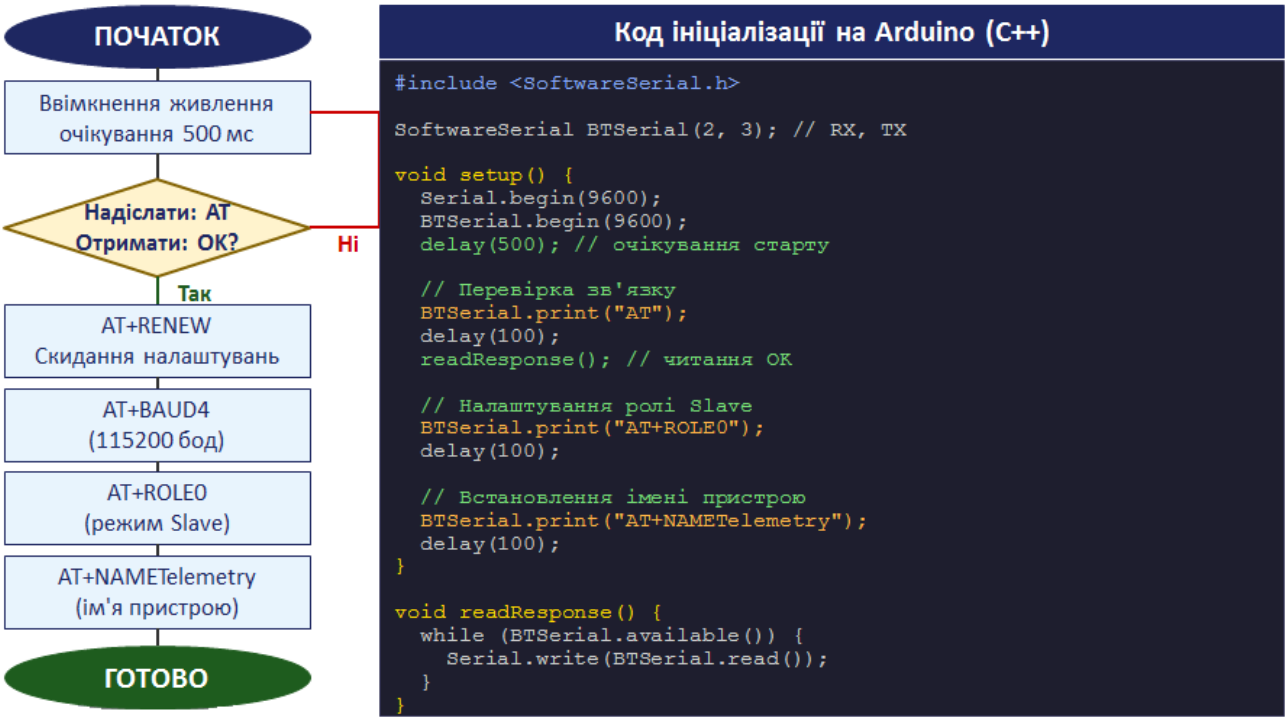


Рисунок 2.10 – Алгоритм ініціалізації модуля HM-10 через AT-команди

Після ввімкнення живлення необхідно витримати затримку не менше 500 мс для завершення внутрішнього завантаження модуля та стабілізації напруги живлення. Потім надсилається тестова команда AT для перевірки зв'язку. Якщо модуль не відповідає, це може свідчити про неправильне підключення ліній TX/RX, невідповідність швидкості UART або несправність модуля. При успішному отриманні відповіді «OK» виконується послідовне налаштування: скидання до заводських налаштувань (AT+RENEW), встановлення швидкості UART (AT+BAUD), встановлення ролі Slave (AT+ROLE0), призначення імені пристрою (AT+NAME).

Наведений на рисунку 2.10 код ініціалізації на C++ для Arduino демонструє практичну реалізацію описаного алгоритму з використанням бібліотеки SoftwareSerial. Бібліотека дозволяє реалізувати програмний UART на виводах D2 та D3, зберігаючи апаратний UART (D0/D1) для відлагодження через монітор порту Arduino IDE. Після завершення процедури ініціалізації модуль готовий до рекламування своєї присутності та очікує підключення від смартфона або іншого Master-пристрою.

Особливості роботи з AT-командами у складі системи телеметрії

При розробці програмного забезпечення системи телеметрії необхідно врахувати ряд практичних особливостей роботи з AT-командами. По-перше, команди налаштування слід виконувати лише при відсутності активного BLE-з'єднання, оскільки в режимі даних модуль ігнорує AT-команди і передає всі вхідні байти напряму через BLE-канал. По-друге, після кожної команди необхідно витримувати затримку не менше 100 мс перед надсиланням наступної для надання модулю часу на обробку та формування відповіді. По-третє, деякі команди (зміна швидкості, зміна ролі) набирають чинності лише після перезапуску модуля командою AT+RESET, тому їх слід використовувати лише при першому налаштуванні або при необхідності зміни конфігурації, а не при кожному увімкненні системи.

Таким чином, UART-інтерфейс у поєднанні з системою AT-команд забезпечує повний контроль над модулем HM-10 та є достатнім для реалізації всіх необхідних функцій системи телеметрії – від початкового налаштування до моніторингу стану з'єднання та передачі телеметричних пакетів у режимі реального часу.

2.4 Вибір та обґрунтування датчиків для збору телеметричних даних

Вибір датчиків є одним із ключових етапів проектування системи телеметрії, оскільки саме датчики визначають перелік вимірюваних фізичних величин, точність отримуваних даних, складність апаратної реалізації та загальне енергоспоживання системи. При виборі датчиків для розроблюваної системи телеметрії керувались такими критеріями: відповідність вимогам промислового застосування, наявність цифрового або аналогового виходу з достатньою роздільною здатністю, сумісність з мікроконтролерною платформою Arduino UNO, доступність та задокументованість компонентів. За результатами аналізу обрано чотири датчики, що охоплюють різні фізичні величини та реалізують три різних інтерфейси підключення: 1-Wire, аналоговий АЦП та I²C. Порівняльна характеристика обраних датчиків наведена на рисунку 2.11.

Датчик	Величина	Діапазон	Точність	Інтерфейс	Живлення	Застосування
DS18B20	Температура	-55...+125 °C	±0.5 °C	1-Wire	3.0-5.5 В	Промисловий моніторинг температури
MPX5700	Тиск	0...700 кПа	±2.5%	Аналог (АЦП)	5.0 В	Гідравліка, пневматика
ADXL345	Прискорення	±2...±16 g	13 біт	I²C / SPI	2.0-3.6 В	Моніторинг вібрації обладнання
INA219	Струм/напруга	0...26 В / 3.2 А	±1%	I²C	3.0-5.5 В	Моніторинг енергоспоживання

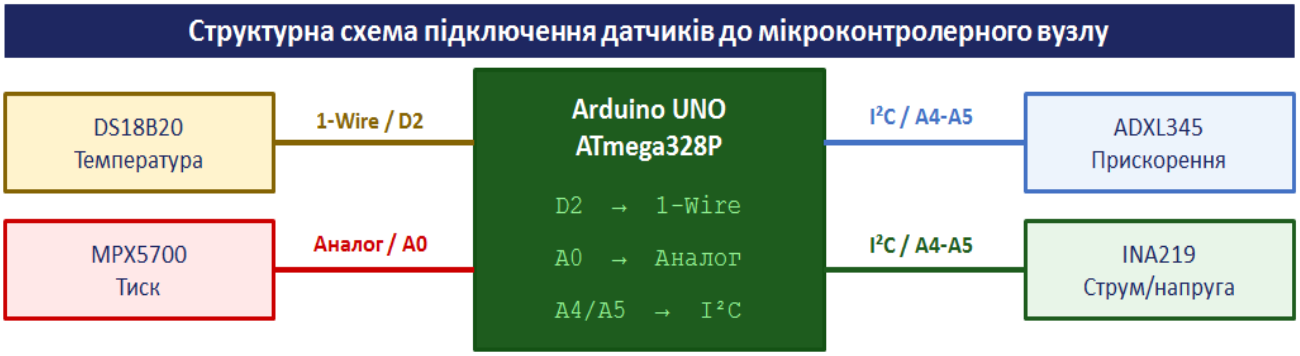


Рисунок 2.11 – Порівняльна характеристика обраних датчиків системи телеметрії

Як видно з рисунку, обраний набір датчиків забезпечує вимірювання температури, тиску, вібрації та електричних параметрів — тобто основних фізичних величин, що характеризують стан промислового обладнання та навколишнього середовища. Різноманітність інтерфейсів підключення дозволяє продемонструвати роботу з усіма основними типами датчиків, що застосовуються у вбудованих системах.

Датчик температури DS18B20

Датчик DS18B20 виробництва компанії Maxim Integrated (нині Texas Instruments) є цифровим термометром промислового класу з інтерфейсом 1-Wire, що дозволяє передавати виміряні значення температури безпосередньо в цифровому вигляді без необхідності в аналого-цифровому перетворенні на стороні мікроконтролера [18]. Датчик виконаний у корпусі TO-92 або у вигляді герметичного зонду у нержавіючому корпусі, що забезпечує його захист від вологи та хімічно агресивних середовищ, рис. 2.12.

Рисунок 2.12 – Датчик DS18B20

Діапазон вимірювання температури складає від -55 до +125 °C, точність в діапазоні від -10 до +85 °C становить ±0.5 °C без будь-якого зовнішнього калібрування. Роздільна здатність програмно налаштовується від 9 до 12 біт, що відповідає кроку вимірювання від 0.5 до 0.0625 °C. Час перетворення

залежить від обраної роздільної здатності: при 9 біт - 93.75 мс, при 12 біт – 750 мс.

Принцип роботи інтерфейсу 1-Wire полягає в тому, що обмін даними між мікроконтролером та датчиком здійснюється по одній сигнальній лінії DQ з підтягуючим резистором 4.7 кОм до лінії живлення. Кожен датчик DS18B20 має унікальний 64-бітний ідентифікатор ROM, що дозволяє підключати до однієї шини 1-Wire необмежену кількість датчиків і адресно зчитувати дані з кожного з них. Датчик також підтримує режим паразитного живлення, при якому він отримує живлення безпосередньо від сигнальної лінії без окремого проводу живлення, що спрощує монтаж у важкодоступних місцях. Для роботи з датчиком в середовищі Arduino використовується бібліотека OneWire у поєднанні з бібліотекою DallasTemperature.

Датчик DS18B20 широко застосовується в промислових системах моніторингу температури трубопроводів, резервуарів, електричних шаф та навколишнього середовища. Його висока точність, широкий діапазон вимірювання та стійкість до завад на сигнальній лінії роблять його оптимальним вибором для систем телеметрії з тривалими лініями зв'язку між датчиком та мікроконтролером.

Датчик тиску MPX5700

Датчик MPX5700 виробництва компанії NXP Semiconductors є п'єзорезистивним датчиком надлишкового тиску з аналоговим виходом напруги, що пропорційна вимірюваному тиску. Датчик виконаний у корпусі 867B з одним або двома портами підключення трубки тиску та розрахований на вимірювання тиску в діапазоні від 0 до 700 кПа [19].

Вихідна напруга датчика лінійно змінюється від 0.2 В при нульовому тиску до 4.7 В при максимальному тиску 700 кПа при напрузі живлення 5 В. Передавальна функція датчика описується виразом: $V_{out} = V_{cc} \times (0.0012858 \times P + 0.04)$, де P – тиск у кПа. Звідси формула перетворення вихідної напруги у значення тиску має вигляд: $P = (V_{out} / V_{cc} - 0.04) / 0.0012858$. Точність

датчика складає ±2.5% від повного діапазону вимірювання, час відгуку - 1.0 мс, що є достатнім для більшості застосувань моніторингу тиску в гідравлічних та пневматичних системах, рис. 2.13.

Рисунок 2.13 – Датчик MPX5700

Підключення датчика до мікроконтролера Arduino здійснюється через аналоговий вхід A0 вбудованого 10-бітного АЦП. При напрузі живлення 5 В роздільна здатність АЦП Arduino становить 4.88 мВ/одиницю, що відповідає роздільній здатності по тиску близько 2.69 кПа на одиницю АЦП. Для підвищення точності вимірювань рекомендується виконувати усереднення кількох послідовних відліків АЦП. Датчик MPX5700 широко застосовується в системах контролю тиску гідравліки, пневматики, систем вентиляції та кондиціонування, а також у медичному обладнанні.

Акселерометр ADXL345

Датчик ADXL345 виробництва компанії Analog Devices є тривісним мікроелектромеханічним акселерометром (MEMS) з цифровим інтерфейсом I²C або SPI. Він вимірює прискорення одночасно по трьох взаємно перпендикулярних осях X, Y та Z у діапазонах ±2g, ±4g, ±8g або ±16g з програмно налаштовуваною роздільною здатністю 13 біт при максимальному діапазоні ±16g, що відповідає роздільній здатності ~0.004g/одиницю, рис. 2.14.

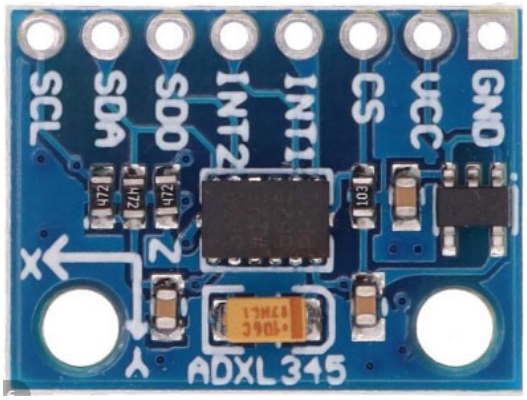


Рисунок 2.14 – Датчик ADXL345

Датчик підтримує програмно налаштовувану частоту вибірки від 0.1 до 3200 Гц, що дозволяє застосовувати його як для виявлення повільних нахилів та орієнтації, так і для аналізу вібрацій обладнання в широкому діапазоні частот. ADXL345 має вбудовані функції виявлення одиночного та подвійного удару, виявлення вільного падіння та перевищення порогового значення прискорення з формуванням апаратного переривання, що дозволяє реалізовувати захисні функції без постійного опитування датчика мікроконтролером [20].

Підключення до Arduino здійснюється через інтерфейс I²C: вивід SDA підключається до аналогового виводу A4, вивід SCL – до A4, що є стандартним апаратним I²C-інтерфейсом мікроконтролера ATmega328P. Адреса датчика на шині I²C визначається станом виводу SDO: при підключенні до GND адреса складає 0x53, при підключенні до VCC - 0x1D. Напруга живлення датчика складає від 2.0 до 3.6 В, тому при підключенні до Arduino з напругою 3.3 В або 5 В необхідно використовувати вбудований стабілізатор 3.3 В плати Arduino. Для роботи з датчиком використовується бібліотека Adafruit ADXL345 або SparkFun ADXL345.

В системі телеметрії датчик ADXL345 виконує функцію моніторингу вібрації та орієнтації обладнання, що є важливим параметром для оцінки стану механічних вузлів – підшипників, насосів, компресорів та інших пристроїв з

обертовими деталями. Підвищений рівень вібрації, як правило, свідчить про знос або несправність механічних компонентів і є важливим сигналом для системи прогностичного технічного обслуговування.

Датчик струму та напруги INA219

Датчик INA219 виробництва компанії Texas Instruments є двонаправленим шунтовим вимірювачем струму та напруги з цифровим інтерфейсом I²C. Принцип роботи полягає у вимірюванні падіння напруги на шунтовому резисторі (у розроблюваній системі - 0.1 Ом) та напруги на шині живлення з подальшим обчисленням струму та потужності у вбудованому обчислювальному блоці з 12-бітним АЦП, рис. 2.15.

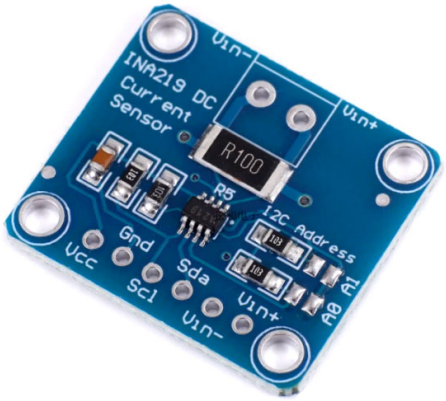


Рисунок 2.15 – Датчик INA219

Датчик забезпечує вимірювання напруги шини у діапазоні від 0 до 26 В та струму до ±3.2 А при шунті 0.1 Ом з точністю ±1% від повного діапазону. Вбудований обчислювальний блок автоматично розраховує потужність з точністю ±0.5%, що звільняє мікроконтролер від необхідності виконання додаткових обчислень. Датчик підтримує програмне налаштування коефіцієнтів калібрування через I²C, що дозволяє адаптувати його до шунтів різного номіналу.

Адреса датчика на шині I²C за замовчуванням складає 0x40 і може бути змінена в діапазоні 0x40–0x4F шляхом підключення виводів A0 та A1 до GND або VCC, що дозволяє підключити до однієї шини I²C до 16 незалежних

датчиків INA219 одночасно. Оскільки і ADXL345, і INA219 використовують інтерфейс I²C, обидва датчики підключаються до однієї шини Arduino (A4=SDA, A5=SCL) при умові відсутності конфлікту адрес. Для роботи з датчиком використовується бібліотека Adafruit INA219.

В системі телеметрії датчик INA219 забезпечує моніторинг енергоспоживання підключеного навантаження у реальному часі, що є важливим параметром як для оцінки ефективності роботи обладнання, так і для контролю стану акумуляторних батарей в автономних системах [21].

Таким чином, обраний набір датчиків DS18B20, MPX5700, ADXL345 та INA219 забезпечує вимірювання чотирьох ключових фізичних параметрів промислового обладнання з використанням трьох різних інтерфейсів підключення. Такий підхід дозволяє продемонструвати універсальність розроблюваної системи телеметрії та її здатність інтегрувати датчики різних типів у єдину вимірювальну мережу з передачею даних через BLE-канал на базі модуля HM-10.

2.5 Функціональна схема системи телеметрії

Функціональна схема є одним із основних конструкторських документів, що відображає склад системи, функції окремих блоків та зв'язки між ними без деталізації конкретної схемотехнічної реалізації. На відміну від принципової електричної схеми, функціональна схема орієнтована на розкриття логіки роботи системи та взаємодії її складових частин, що є важливим для розуміння загальної архітектури розроблюваної системи телеметрії. Загальна функціональна схема системи наведена на рисунку 2.16.

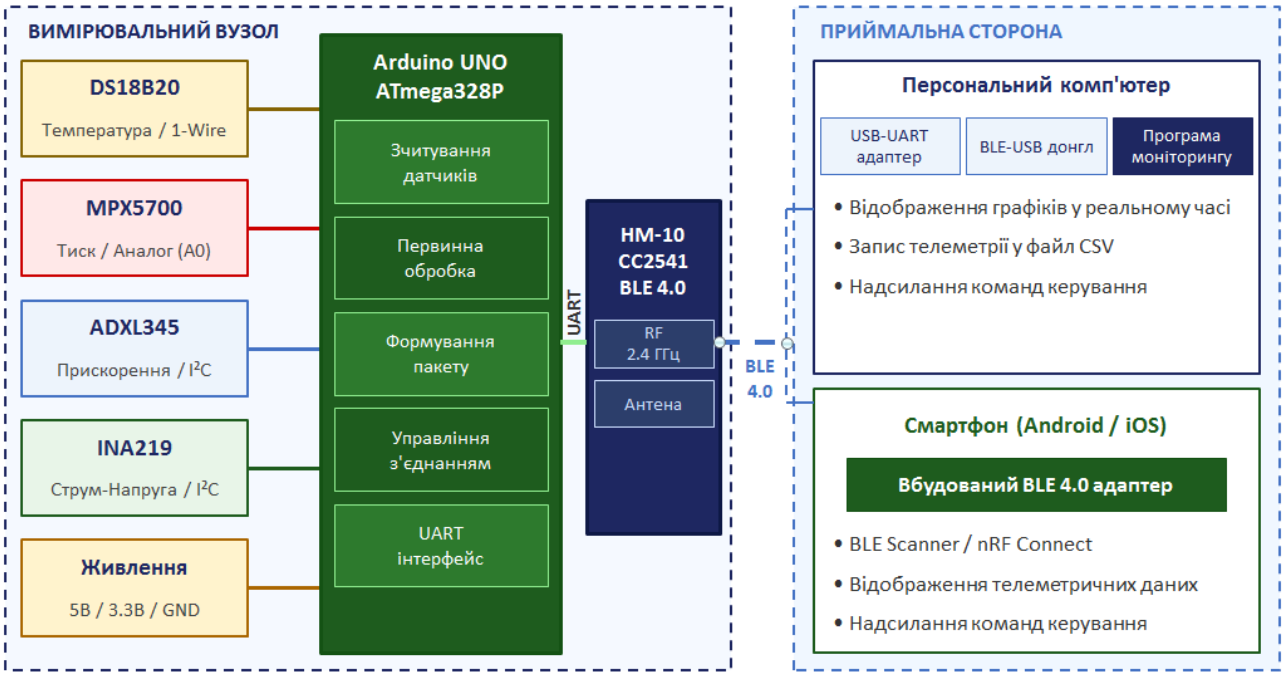


Рисунок 2.16 - Загальна функціональна схема системи телеметрії

Як видно зі схеми, система телеметрії складається з двох основних частин: вимірювального вузлу та приймальної сторони, з'єднаних каналом бездротового зв'язку BLE 4.0. Вимірювальний вузол виконує функції збору даних з датчиків, їх первинної обробки та передачі через BLE-канал, тоді як приймальна сторона забезпечує прийом, відображення та збереження телеметричних даних [9,12].

Вимірювальний вузол

Вимірювальний вузол є центральним функціональним елементом системи та складається з мікроконтролерної плати Arduino UNO, чотирьох датчиків фізичних величин, модуля бездротового зв'язку HM-10 та блоку живлення. Детальна функціональна схема вимірювального вузлу наведена на рисунку 2.17.

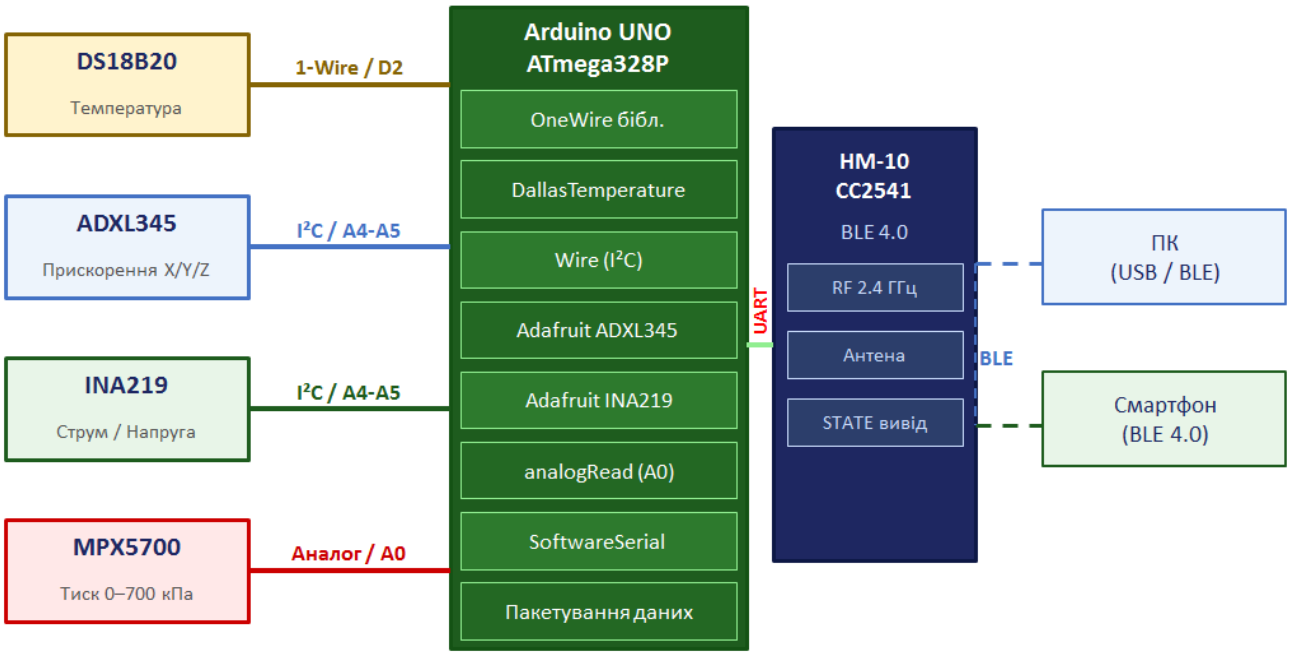


Рисунок 2.17 - Функціональна схема вимірювального вузлу системи телеметрії

Мікроконтролер Arduino UNO на базі ATmega328P виконує роль центрального обчислювального елемента вузлу і реалізує такі функції: циклічне зчитування даних з датчиків через відповідні інтерфейси, первинну обробку отриманих даних (масштабування, фільтрація, усереднення), формування телеметричного пакету фіксованого формату та його передачу до модуля HM-10 через програмний послідовний порт UART. Крім того, мікроконтролер здійснює моніторинг стану BLE-з'єднання через вивід STATE модуля HM-10 та управляє процесом підключення.

Датчики підключаються до мікроконтролера через три різних інтерфейси, що формують відповідні функціональні шини вимірювального вузлу. Шина 1-Wire на виводі D2 забезпечує зв'язок з датчиком температури DS18B20. Шина I²C на виводах A4 (SDA) та A5 (SCL) об'єднує акселерометр ADXL345 та датчик струму і напруги INA219 — обидва пристрої мають різні адреси на шині (0x53 та 0x40 відповідно), що виключає конфлікт при одночасній роботі. Аналоговий вхід A0 підключений до виходу датчика тиску MPX5700, напруга якого перетворюється вбудованим 10-бітним АЦП

мікроконтролера у цифровий код для подальших обчислень. Шина UART на виводах D2 (RX) та D3 (TX), реалізована за допомогою бібліотеки SoftwareSerial, забезпечує зв'язок між мікроконтролером та модулем HM-10. Шина живлення постачає напругу 5 В та 3.3 В від блоку живлення (USB або акумулятор) до всіх компонентів вузлу [22].

Канал бездротового зв'язку

Канал бездротового зв'язку реалізований на основі модуля HM-10 з мікроконтролером CC2541, що забезпечує прозору передачу байтових послідовностей між вимірювальним вузлом та пристроєм-приймачем через BLE 4.0 на частоті 2.4 ГГц. Модуль функціонує в ролі периферійного пристрою (Peripheral/Slave) та очікує підключення від центрального пристрою (Central) — смартфона або BLE-адаптера ПК. Після встановлення з'єднання всі байти, що надходять до модуля по UART від мікроконтролера, автоматично передаються по BLE-каналу до підключеного приймача і навпаки.

Приймальна сторона

Приймальна сторона системи може бути реалізована у двох варіантах: персональний комп'ютер або мобільний смартфон. У варіанті з персональним комп'ютером підключення здійснюється через USB-UART адаптер (якщо ПК підключено кабелем до Arduino) або через BLE-USB донгл (якщо використовується бездротове підключення безпосередньо до модуля HM-10). Програма моніторингу на ПК забезпечує прийом та декодування телеметричних пакетів, відображення поточних значень параметрів та їх динаміки у вигляді графіків, а також запис даних у файл формату CSV для подальшого аналізу. У варіанті зі смартфоном підключення здійснюється безпосередньо через вбудований BLE-адаптер мобільного пристрою за допомогою стандартних додатків або спеціалізованого мобільного застосунку.

Алгоритм роботи та формат телеметричного пакету

Алгоритм роботи вимірювального вузлу та формат телеметричного пакету наведені на рисунку 2.18.

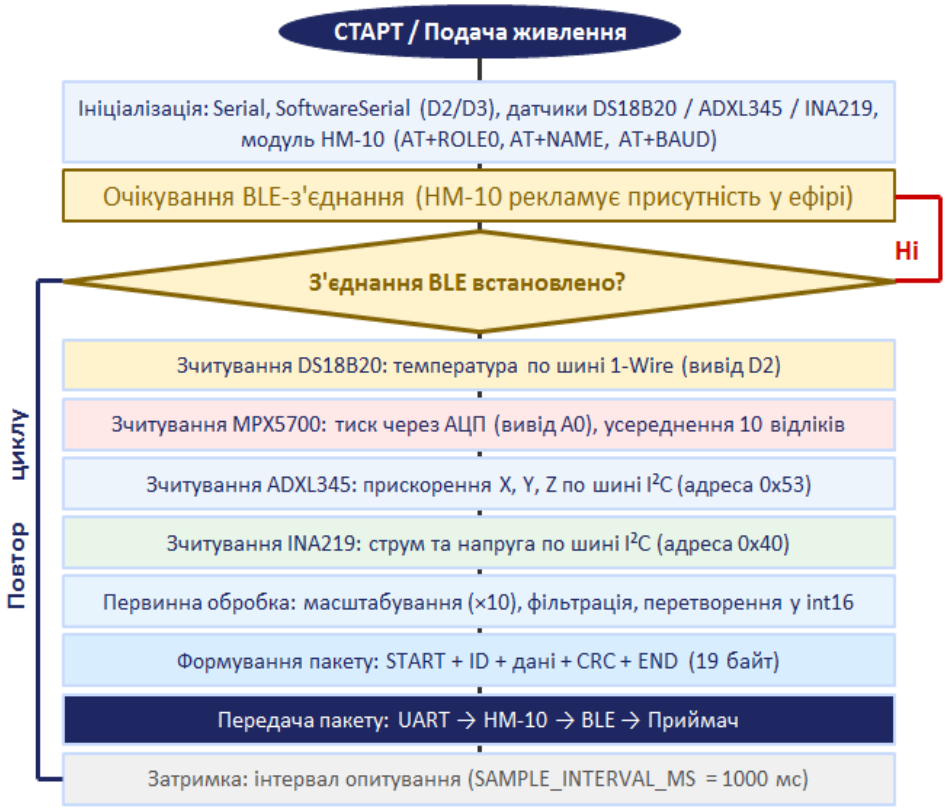


Рисунок 2.18 - Алгоритм роботи системи телеметрії

Після подачі живлення мікроконтролер виконує ініціалізацію всіх підключених датчиків та модуля HM-10 через послідовність AT-команд. Далі система переходить у режим очікування BLE-з'єднання — модуль HM-10 рекламує свою присутність у ефірі. Після встановлення з'єднання запускається основний цикл збору та передачі телеметричних даних, що повторюється з заданим інтервалом опитування (за замовчуванням 1 секунда).

Телеметричний пакет має фіксований формат загальним розміром 19 байт. Пакет починається маркером початку (0xAA) та ідентифікатором вузла, після чого розміщуються поля даних від кожного датчика: температура (2 байти, значення × 10), тиск (2 байти, кПа × 10), прискорення по осях X, Y, Z (по 2 байти кожне, значення в mg), напруга шини (2 байти, мВ) та струм (2 байти, мА × 10). Завершується пакет байтом контрольної суми (XOR всіх попередніх байт) та маркером кінця (0x55). Використання цілочисельних значень з фіксованим масштабом замість чисел з плаваючою комою дозволяє

суттєво спростити формування та розбір пакету на обох кінцях каналу зв'язку, зменшити обсяг пакету та підвищити надійність передачі. Максимальна частота передачі пакетів при даному форматі становить близько 10 Гц, що є достатнім для всіх розглянутих датчиків.

Таким чином, функціональна схема системи телеметрії чітко відображає розподіл функцій між компонентами системи, логіку їх взаємодії та маршрути проходження даних від датчиків до пристрою відображення. Розроблена схема є основою для подальшої реалізації програмного забезпечення системи та забезпечує однозначне розуміння принципів роботи кожного функціонального блоку.

2.6 Архітектура програмного забезпечення та програмування мікроконтролерного вузла

Програмне забезпечення системи телеметрії є невід'ємною складовою, що реалізує логіку збору, обробки та передачі даних. При проектуванні програмного забезпечення розроблюваної системи дотримувались принципів модульності, розшарування відповідальності та мінімальної зв'язності між компонентами. Архітектура побудована за принципом розшарування — кожен рівень взаємодіє виключно з безпосередньо сусідніми рівнями. Програма мікроконтролерного вузлу містить п'ять рівнів: апаратний рівень, рівень апаратних інтерфейсів, рівень драйверів датчиків, рівень обробки даних та прикладний рівень. Між мікроконтролерним вузлом та приймальною стороною розташований BLE-канал на базі модуля HM-10, що забезпечує прозору двонаправлену передачу даних [18, 23].

Ініціалізація системи

Програма мікроконтролерного вузлу розроблена у середовищі Arduino IDE мовою C++. Функція setup() виконується один раз при запуску та здійснює

ініціалізацію всіх підсистем — послідовних портів, датчиків та модуля HM-10. На початку файлу підключаються необхідні бібліотеки та оголошуються глобальні об'єкти для роботи з периферією.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <Adafruit_ADXL345_U.h>
#include <Adafruit_INA219.h>
#include <SoftwareSerial.h>
// ── Конфігурація виводів та констант ──────────────────────────
#define ONE_WIRE_BUS  2     // DS18B20 — вивід D2
#define BT_RX_PIN     2     // HM-10 TX → Arduino D2
#define BT_TX_PIN     3     // HM-10 RX ← Arduino D3
#define PRESSURE_PIN  A0    // MPX5700 — аналоговий вхід A0
#define SAMPLE_MS     1000  // інтервал опитування, мс
#define PKT_SIZE      19    // розмір пакету, байт
#define NODE_ID       0x01  // ідентифікатор вузла
// ── Об'єкти периферії ─────────────────────────────────────────
OneWire          oneWire(ONE_WIRE_BUS);
DallasTemperature ds18b20(&oneWire);
Adafruit_ADXL345_Unified accel(12345);
Adafruit_INA219   ina219;
SoftwareSerial    BTSerial(BT_RX_PIN, BT_TX_PIN);
bool bleConnected = false; // прапорець стану BLE-з'єднання
void setup() {
  Serial.begin(9600);   // порт відлагодження (D0/D1)
  BTSerial.begin(9600); // зв'язок з HM-10 (D2/D3)
  // Ініціалізація датчиків
  ds18b20.begin();
  if (!accel.begin())  Serial.println("ADXL345 не знайдено!");
  if (!ina219.begin()) Serial.println("INA219 не знайдено!");
  // Налаштування модуля HM-10 через AT-команди
  delay(500);                    // очікування старту модуля
  BTSerial.print("AT");        delay(100); // тест зв'язку
  BTSerial.print("AT+ROLE0");  delay(100); // режим Slave
  BTSerial.print("AT+NAMETelemetry"); delay(100); // ім'я пристрою
  BTSerial.print("AT+NOTI1");  delay(100); // сповіщення про з'єднання
  Serial.println("Ініціалізація завершена. Очікування BLE...");
}
```

Бібліотека SoftwareSerial дозволяє реалізувати програмний UART на виводах D2 та D3, залишаючи апаратний UART (D0/D1) вільним для відлагодження через монітор порту Arduino IDE. Команда AT+NOTI1 вмикає режим сповіщень модуля HM-10 — при встановленні з'єднання модуль надсилає по UART рядок «OK+CONN», а при розриві — «OK+LOST». Це

дозволяє програмі відстежувати стан з'єднання без постійного опитування виводу STATE.

Головний цикл збору та передачі даних

Функція loop() реалізує основний цикл роботи системи. На початку кожної ітерації перевіряється стан BLE-з'єднання через надходження сповіщень від модуля HM-10. Якщо з'єднання не встановлено — цикл пропускається і система продовжує очікування.

```
void loop() {
 // Перевірка сповіщень від HM-10 про стан з'єднання
 if (BTSerial.available()) {
   String msg = BTSerial.readStringUntil('\n');
   if (msg.indexOf("OK+CONN") >= 0) {
     bleConnected = true;
     Serial.println("BLE: з'єднання встановлено");   }
   if (msg.indexOf("OK+LOST") >= 0) {
     bleConnected = false;
     Serial.println("BLE: з'єднання розірвано");  }  }
 // Якщо BLE-з'єднання відсутнє — передачу не виконуємо
 if (!bleConnected) return;
 // ── Зчитування даних з усіх датчиків ──────────────────────
 float   temp = readTemperature();
 float   pres = readPressure();
 int16_t ax, ay, az;
 readAccel(&ax, &ay, &az);
 float volt = ina219.getBusVoltage_V();
 float curr = ina219.getCurrent_mA();
 // ── Вивід у монітор порту для відлагодження ───────────────
 Serial.print("T="); Serial.print(temp);
 Serial.print(" P="); Serial.print(pres);
 Serial.print(" AX="); Serial.print(ax);
 Serial.print(" V="); Serial.print(volt);
 Serial.print(" I="); Serial.println(curr);
 // ── Формування та відправка пакету ────────────────────────
 uint8_t pkt[PKT_SIZE];
 buildPacket(pkt, temp, pres, ax, ay, az, volt, curr);
 BTSerial.write(pkt, PKT_SIZE);
 delay(SAMPLE_MS); // затримка до наступного циклу}
```

Виведення проміжних значень у монітор порту через Serial.print() є важливим інструментом відлагодження – він дозволяє контролювати коректність зчитування датчиків без необхідності використання BLE-

з'єднання. При розгортанні готової системи ці виклики можна відключити директивою препроцесора для зменшення навантаження на мікроконтролер.

Зчитування температури (DS18B20)

Датчик DS18B20 використовує протокол 1-Wire. Метод requestTemperatures() ініціює вимірювання на всіх датчиках шини, після чого getTempCByIndex(0) повертає результат першого знайденого датчика у градусах Цельсія. При роздільній здатності 12 біт час перетворення складає 750 мс, тому виклик requestTemperatures() розміщений на початку циклу, а зчитування результату – після виконання інших операцій.

```
float readTemperature() {
 ds18b20.requestTemperatures(); // запит вимірювання (750 мс)
 float t = ds18b20.getTempCByIndex(0); // зчитування результату
 if (t == DEVICE_DISCONNECTED_C) {
  Serial.println("Помилка: DS18B20 не відповідає");
  return -999.0; // код помилки  }
 return t; // температура у °C}
```

Зчитування тиску (MPX5700)

Датчик MPX5700 має аналоговий вихід. Вбудований 10-бітний АЦП Arduino перетворює напругу від 0 до 5 В у значення від 0 до 1023. Для зменшення впливу шумів АЦП виконується усереднення 10 послідовних відліків з паузою 2 мс між ними. Перетворення відліку АЦП у фізичний тиск виконується за передавальною функцією датчика.

```
float readPressure() {
 // Усереднення 10 відліків для зменшення шуму АЦП
 long sum = 0;
 for (int i = 0; i < 10; i++) {
  sum += analogRead(PRESSURE_PIN);
  delay(2);    }
 float adcVal = sum / 10.0;
 // Перетворення відліку АЦП → напруга → тиск
 float vout = (adcVal / 1023.0) * 5.0;          // В
 float pres = (vout / 5.0 - 0.04) / 0.0012858;  // кПа
 return pres;  }
```

Зчитування прискорення (ADXL345)

Акселерометр ADXL345 підключений по шині I²C. Бібліотека Adafruit ADXL345 повертає прискорення у м/с², тому для передачі у пакеті значення

перетворюються у мілі-g (mg), де 1g = 9.81 м/с². Тип int16_t дозволяє зберігати значення від -32768 до +32767 mg, що перекриває весь діапазон датчика ±16g (±16000 mg).

```
void readAccel(int16_t *ax, int16_t *ay, int16_t *az) {
  sensors_event_t event;
  accel.getEvent(&event);
  // Перетворення м/с² → мілі-g
  *ax = (int16_t)(event.acceleration.x / 9.81 * 1000);
  *ay = (int16_t)(event.acceleration.y / 9.81 * 1000);
  *az = (int16_t)(event.acceleration.z / 9.81 * 1000);}
```

Зчитування струму та напруги (INA219)

Датчик INA219 підключений по тій самій шині I²C що і ADXL345, але має іншу адресу (0x40). Бібліотека Adafruit INA219 надає зручні методи для безпосереднього отримання напруги та струму у фізичних одиницях – getBusVoltage_V() повертає напругу шини у вольтах, getCurrent_mA() - струм у міліамперах. Додаткова функція обчислення потужності показана у лістингу

```
void readPower(float *voltage, float *current, float *power) {
  *voltage = ina219.getBusVoltage_V();
  *current = ina219.getCurrent_mA();
  *power   = (*voltage) * (*current) / 1000.0; // Вт
  // Перевірка на від'ємний струм (зарядка акумулятора)
  if (*current < 0) *current = 0;
  Serial.print("U="); Serial.print(*voltage); Serial.print("V ");
  Serial.print("I="); Serial.print(*current); Serial.print("mA ");
  Serial.print("P="); Serial.print(*power);   Serial.println("W");}
```

Формування телеметричного пакету

Функція buildPacket() формує пакет фіксованого формату розміром 19 байт. Для зберігання значень з одним знаком після коми у цілочисельному форматі застосовується масштабування на коефіцієнт 10. Двобайтові значення записуються у порядку Little-Endian (молодший байт першим). Контрольна сума обчислюється методом XOR усіх байт пакету від 0 до 16 включно.

```
void buildPacket(uint8_t *pkt, float temp, float pres,
                 int16_t ax, int16_t ay, int16_t az,
                 float volt, float curr) {
  // Масштабування значень × 10 → int16
  int16_t iTemp = (int16_t)(temp * 10);   // 24.5°C → 245
```

```
  int16_t iPres = (int16_t)(pres * 10);   // 101.3 кПа → 1013
  int16_t iVolt = (int16_t)(volt * 1000); // 3.3 В → 3300 мВ
  int16_t iCurr = (int16_t)(curr * 10);   // 12.5 мА → 125
  pkt[0]  = 0xAA;         // START
  pkt[1]  = NODE_ID;      // ID вузла
  // Поля даних (Little-Endian)
  pkt[2]  = iTemp & 0xFF;  pkt[3]  = (iTemp >> 8) & 0xFF;
  pkt[4]  = iPres & 0xFF;  pkt[5]  = (iPres >> 8) & 0xFF;
  pkt[6]  = ax    & 0xFF;  pkt[7]  = (ax    >> 8) & 0xFF;
  pkt[8]  = ay    & 0xFF;  pkt[9]  = (ay    >> 8) & 0xFF;
  pkt[10] = az    & 0xFF;  pkt[11] = (az    >> 8) & 0xFF;
  pkt[12] = iVolt & 0xFF;  pkt[13] = (iVolt >> 8) & 0xFF;
  pkt[14] = iCurr & 0xFF;  pkt[15] = (iCurr >> 8) & 0xFF;
  // CRC — XOR байтів 0…15
  uint8_t crc = 0;
  for (int i = 0; i < 16; i++) crc ^= pkt[i];
  pkt[16] = crc;
  pkt[17] = 0x00; // зарезервовано
  pkt[18] = 0x55; // END}
// Функція декодування пакету (для відлагодження)
bool parsePacket(uint8_t *pkt) {
  if (pkt[0] != 0xAA || pkt[18] != 0x55) return false; // перевірка маркерів
  uint8_t crc = 0;
  for (int i = 0; i < 16; i++) crc ^= pkt[i];
  if (crc != pkt[16]) return false; // перевірка CRC
  // Розпакування значень
  float temp = ((int16_t)(pkt[3] << 8 | pkt[2])) / 10.0;
  float pres = ((int16_t)(pkt[5] << 8 | pkt[4])) / 10.0;
  Serial.print("Розпакований пакет: T=");
  Serial.print(temp); Serial.print("°C P=");
  Serial.print(pres); Serial.println("кПа");
  return true;}
```

Наявність функції parsePacket() є важливою для відлагодження системи – вона дозволяє перевіряти коректність формування пакету безпосередньо на стороні мікроконтролера шляхом формування пакету та його негайного розпакування і виводу через Serial. Ця функція також є основою для реалізації аналогічної процедури декодування на стороні програми-клієнта для ПК.

Таким чином, програмне забезпечення мікроконтролерного вузлу реалізоване у вигляді набору чітко розмежованих функцій, кожна з яких відповідає за одну конкретну задачу. Використання перевірених бібліотек Arduino для роботи з датчиками дозволяє зосередитись на логіці системи телеметрії. Наведені лістинги повністю відображають ключові алгоритмічні

рішення, прийняті при розробці програмного забезпечення вимірювального вузлу.

2.7 Протокол передачі даних та розроблення програми-клієнта для ПК

Програма-клієнт для персонального комп'ютера є кінцевою ланкою системи телеметрії, що забезпечує прийом телеметричних пакетів від модуля HM-10 через BLE-канал, їх декодування, відображення у зручному для оператора вигляді та збереження у файл для подальшого аналізу [24]. Програма розроблена мовою Python з використанням бібліотек pyserial для роботи з послідовним портом, matplotlib для побудови графіків та tkinter для графічного інтерфейсу користувача. Архітектура програми та протокол передачі даних наведені на рисунку 2.19.

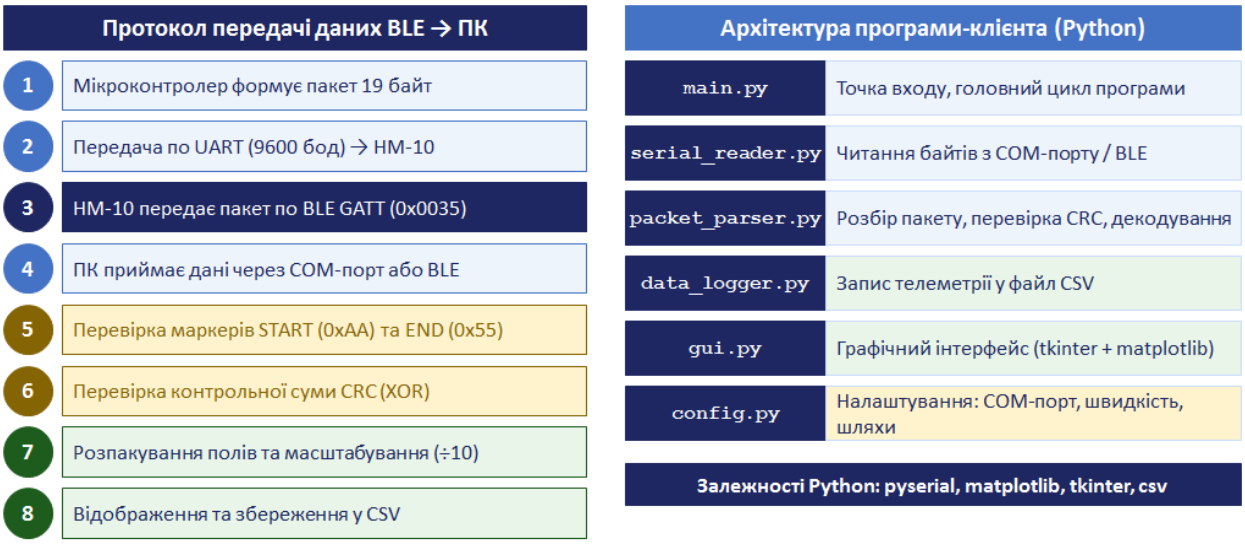


Рисунок 2.19 – Архітектура програми-клієнта для ПК та протокол передачі даних

Протокол передачі даних

Передача телеметричних даних від мікроконтролерного вузлу до ПК відбувається у вісім кроків. Мікроконтролер Arduino формує пакет фіксованого розміру 19 байт та надсилає його по UART зі швидкістю 9600 бод

до модуля HM-10. Модуль HM-10 прозоро передає прийняті байти через BLE-канал по GATT-характеристиці з UUID 0x0035. Програма на ПК приймає дані через COM-порт при підключенні через USB-UART адаптер або через BLE-стек операційної системи Windows при підключенні через BLE-USB донгл. Після прийому байтів програма виконує перевірку наявності маркерів початку (0xAA) та кінця (0x55) пакету, перевірку контрольної суми CRC методом XOR, розпакування полів даних та масштабування (ділення на 10 для відновлення значень з одним знаком після коми) і виведення на екран та збереження у CSV-файл.

Структура програми-клієнта

Програма організована у вигляді шести модулів відповідно до рисунку 2.19. Модуль config.py містить усі налаштовувані параметри системи — ім'я COM-порту, швидкість UART, ідентифікатор вузла та шляхи до файлів збереження. Винесення конфігурації до окремого файлу дозволяє адаптувати програму до різних апаратних конфігурацій без зміни основного коду. Лістинг показує структуру конфігураційного файлу.

python
```
# config.py — налаштування програми-клієнта
# Параметри послідовного порту
COM_PORT    = "COM5"   # Диспетчер пристроїв Windows → Порти (COM і LPT)
BAUD_RATE   = 9600
TIMEOUT     = 1.0      # с
# Параметри пакету
PKT_SIZE    = 19
PKT_START   = 0xAA
PKT_END     = 0x55
NODE_ID     = 0x01
# Збереження даних
LOG_FILE    = "telemetry_log.csv"
MAX_POINTS  = 60       # кількість точок на графіку
UPDATE_MS   = 100      # інтервал оновлення GUI, мс
```

Читання даних з COM-порту

Модуль serial_reader.py реалізує потоковий прийом байтів з послідовного порту. Читання виконується у окремому потоці (Thread), щоб не

блокувати графічний інтерфейс. Буфер накопичує вхідні байти та виділяє з них повні пакети за маркерами. Лістинг показує реалізацію модуля читання.

```
python
import serial, threading, queue
from config import COM_PORT, BAUD_RATE, TIMEOUT, PKT_SIZE, PKT_START,
PKT_END
class SerialReader:
    def __init__(self):
        self.port      = None
        self.running   = False
        self.pkt_queue = queue.Queue()  # черга готових пакетів
        self._buf      = bytearray()
    def start(self):
        """Відкрити порт та запустити потік читання"""
        self.port    = serial.Serial(COM_PORT, BAUD_RATE, timeout=TIMEOUT)
        self.running = True
        self._thread = threading.Thread(target=self._read_loop, daemon=True)
        self._thread.start()
        print(f"Підключено до {COM_PORT} @ {BAUD_RATE} бод")
    def stop(self):
        self.running = False
        if self.port and self.port.is_open:
            self.port.close()
    def _read_loop(self):
        """Фоновий потік: читання байтів та пошук пакетів"""
        while self.running:
            try:
                data = self.port.read(self.port.in_waiting or 1)
                if data:
                    self._buf.extend(data)
                    self._extract_packets()
            except serial.SerialException as e:
                print(f"Помилка порту: {e}")
                self.running = False
    def _extract_packets(self):
        """Виділення пакетів з буфера за маркерами START/END"""
        while len(self._buf) >= PKT_SIZE:
            idx = self._buf.find(PKT_START)
            if idx == -1:
                self._buf.clear()
                return
            if idx > 0:
                del self._buf[:idx]
            if len(self._buf) >= PKT_SIZE:
                if self._buf[PKT_SIZE - 1] == PKT_END:
                    pkt = bytes(self._buf[:PKT_SIZE])
                    self.pkt_queue.put(pkt)
                del self._buf[:PKT_SIZE]
```

Використання черги (queue.Queue) для передачі пакетів між потоком читання та головним потоком GUI забезпечує потокобезпечний обмін даними без необхідності явної синхронізації через м'ютекси.

Розбір та верифікація пакету

Модуль packet_parser.py відповідає за декодування прийнятого пакету, перевірку його цілісності та повернення структурованих даних. Лістинг показує реалізацію парсеру пакетів.

```
python
import struct
from config import PKT_START, PKT_END
def parse_packet(pkt: bytes) -> dict | None:
    """
    Розібрати телеметричний пакет.
    Повертає dict з даними або None при помилці.
    """
    # Перевірка маркерів
    if pkt[0] != PKT_START or pkt[18] != PKT_END:
        print("Помилка: невірні маркери пакету")
        return None
    # Перевірка CRC (XOR байтів 0..15)
    crc_calc = 0
    for i in range(16):
        crc_calc ^= pkt[i]
    if crc_calc != pkt[16]:
        print(f"Помилка CRC: очікувалось {pkt[16]:02X}, отримано {crc_calc:02X}")
        return None
    # Розпакування полів (Little-Endian, знакові int16)
    _, node_id, \
    raw_temp, raw_pres, \
    ax, ay, az, \
    raw_volt, raw_curr, \
    crc, _ = struct.unpack('<BBhhhhhhhBB', pkt[:18] + b'\x00')
    # Масштабування до фізичних одиниць
    return {
        "node_id":  node_id,
        "temp":     raw_temp  / 10.0,   # °C
        "pressure": raw_pres  / 10.0,   # кПа
        "accel_x":  ax,                 # мg
        "accel_y":  ay,                 # мg
        "accel_z":  az,                 # мg
        "voltage":  raw_volt  / 1000.0, # В
        "current":  raw_curr  / 10.0,   # мА
        "power":    (raw_volt / 1000.0) * (raw_curr / 10.0) / 1000.0,  # Вт  }
```

Використання модуля struct зі специфікатором формату '<BBhhhhhhhBB' дозволяє компактно описати структуру пакету та розпакувати всі поля одним викликом. Символ '<' вказує на порядок байт Little-Endian, 'B' — беззнаковий байт, 'h' — знакове 16-бітне ціле число.

Збереження телеметрії у CSV-файл

Модуль data_logger.py відповідає за запис прийнятих даних у файл CSV для подальшого аналізу в Microsoft Excel або спеціалізованих інструментах аналізу даних. Лістинг показує реалізацію логера.

```
python
import csv, os
from datetime import datetime
from config import LOG_FILE
class DataLogger:
    FIELDS = ["timestamp", "node_id", "temp", "pressure",
              "accel_x", "accel_y", "accel_z",
              "voltage", "current", "power"]
    def __init__(self):
        self._file   = None
        self._writer = None
        self.count   = 0
    def open(self, filename: str = LOG_FILE):
        """Відкрити файл та записати заголовок"""
        write_header = not os.path.exists(filename)
        self._file   = open(filename, "a", newline="", encoding="utf-8")
        self._writer = csv.DictWriter(self._file, fieldnames=self.FIELDS)
        if write_header:
            self._writer.writeheader()
        print(f"Лог відкрито: {filename}")
    def log(self, data: dict):
        """Записати рядок телеметрії з міткою часу"""
        row = {"timestamp": datetime.now().isoformat()} | data
        self._writer.writerow(row)
        self._file.flush()  # негайний запис на диск
        self.count += 1
    def close(self):
        if self._file:
            self._file.close()
            print(f"Лог закрито. Записано рядків: {self.count}")
```

Графічний інтерфейс користувача

Головний модуль програми реалізує графічний інтерфейс на основі бібліотеки tkinter з вбудованим полотном matplotlib для відображення графіків у реальному часі. Інтерфейс програми-клієнта наведено на рисунку 2.20.

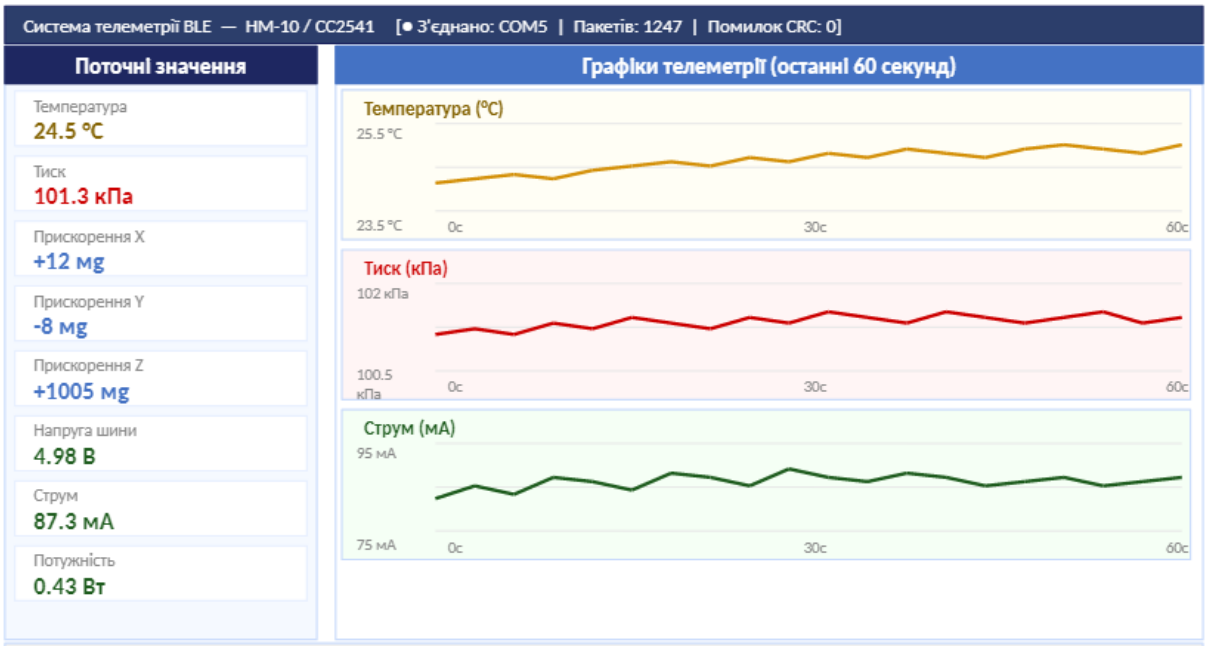


Рисунок 2.20 - Інтерфейс програми-клієнта для відображення телеметричних даних

Лістинг показує реалізацію головного циклу програми з оновленням графіків.

```
python
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
from collections import deque
from serial_reader import SerialReader
from packet_parser import parse_packet
from data_logger import DataLogger
from config import MAX_POINTS, UPDATE_MS
class TelemetryApp:
    def __init__(self, root):
        self.root    = root
        self.root.title("Система телеметрії BLE — HM-10 / CC2541")
        self.reader  = SerialReader()
        self.logger  = DataLogger()
        # Буфери даних для графіків (кільцеві черги)
        self.buf_temp = deque([0.0] * MAX_POINTS, maxlen=MAX_POINTS)
        self.buf_pres = deque([0.0] * MAX_POINTS, maxlen=MAX_POINTS)
        self.buf_curr = deque([0.0] * MAX_POINTS, maxlen=MAX_POINTS)
        self.pkt_count = 0
        self.err_count = 0
        self._build_ui()
        self._build_charts()
    def _build_ui(self):
        """Побудова панелі поточних значень та кнопок"""
        left = tk.Frame(self.root, width=200, bg="#F5F9FF")
        left.pack(side=tk.LEFT, fill=tk.Y, padx=5, pady=5)
```

```
        self.lbl_temp = tk.Label(left, text="— °C",  font=("Arial", 18, "bold"), fg="#856404",
bg="#F5F9FF")
        self.lbl_pres = tk.Label(left, text="— кПа", font=("Arial", 18, "bold"), fg="#CC0000",
bg="#F5F9FF")
        self.lbl_volt = tk.Label(left, text="— В",   font=("Arial", 18, "bold"), fg="#1E5C1E",
bg="#F5F9FF")
        self.lbl_curr = tk.Label(left, text="— мА",  font=("Arial", 18, "bold"), fg="#1E5C1E",
bg="#F5F9FF")
        for lbl in [self.lbl_temp, self.lbl_pres, self.lbl_volt, self.lbl_curr]:
            lbl.pack(pady=8, padx=10, anchor="w")
        tk.Button(left, text="▶ Старт",     bg="#1E5C1E", fg="white",
                  command=self.start).pack(fill=tk.X, padx=5, pady=2)
        tk.Button(left, text="■ Стоп",      bg="#CC0000",  fg="white",
                  command=self.stop).pack(fill=tk.X, padx=5, pady=2)
        tk.Button(left, text="□ Зберегти", bg="#4472C4",  fg="white",
                  command=self.save).pack(fill=tk.X, padx=5, pady=2)
    def _build_charts(self):
        """Створення полотна matplotlib"""
        self.fig, (self.ax1, self.ax2, self.ax3) = plt.subplots(3, 1, figsize=(8, 6))
        self.fig.tight_layout(pad=2.0)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.root)
        self.canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)
    def _update(self):
        """Цикл оновлення — виклик кожні UPDATE_MS мс"""
        while not self.reader.pkt_queue.empty():
            raw_pkt = self.reader.pkt_queue.get()
            data    = parse_packet(raw_pkt)
            if data:
                self.pkt_count += 1
                self.logger.log(data)
                self._refresh_labels(data)
                self.buf_temp.append(data["temp"])
                self.buf_pres.append(data["pressure"])
                self.buf_curr.append(data["current"])
            else:
                self.err_count += 1
        # Оновлення графіків
        for ax, buf, title, color in [
            (self.ax1, self.buf_temp, "Температура (°C)", "#D4910A"),
            (self.ax2, self.buf_pres, "Тиск (кПа)",       "#CC0000"),
            (self.ax3, self.buf_curr, "Струм (мА)",       "#1E5C1E"),
        ]:
            ax.clear()
            ax.plot(list(buf), color=color, linewidth=1.5)
            ax.set_title(title, fontsize=9)
            ax.grid(True, alpha=0.3)
        self.canvas.draw()
        self.root.title(
            f"Телеметрія BLE  |  Пакетів: {self.pkt_count}  |  Помилок CRC: {self.err_count}"    )
        self.root.after(UPDATE_MS, self._update)
```

```
    def _refresh_labels(self, d):
        self.lbl_temp.config(text=f"{d['temp']:.1f} °C")
        self.lbl_pres.config(text=f"{d['pressure']:.1f} кПа")
        self.lbl_volt.config(text=f"{d['voltage']:.2f} В")
        self.lbl_curr.config(text=f"{d['current']:.1f} мА")
    def start(self):
        self.reader.start()
        self.logger.open()
        self._update()
    def stop(self):
        self.reader.stop()
        self.logger.close()
    def save(self):
        print(f"Збережено {self.logger.count} записів у {self.logger._file.name}")
if __name__ == "__main__":
    root = tk.Tk()
    app  = TelemetryApp(root)
    root.mainloop()
```

Метод root.after(UPDATE_MS, self._update) є ключовим елементом архітектури GUI — він планує повторний виклик функції оновлення через заданий інтервал у головному циклі подій tkinter, що забезпечує неблокуюче оновлення інтерфейсу без необхідності використання додаткових потоків для GUI. Використання кільцевих черг deque з обмеженою довжиною (maxlen=MAX_POINTS) автоматично видаляє найстаріші значення при додаванні нових, забезпечуючи відображення лише останніх MAX_POINTS секунд телеметрії.

Запуск та розгортання програми

Для встановлення залежностей та запуску програми необхідно виконати такі команди у командному рядку Windows:

```
bash
# Встановлення необхідних бібліотек
pip install pyserial matplotlib
# Запуск програми-клієнта
python main.py
```

Перед запуском у файлі config.py необхідно вказати правильне ім'я COM-порту, до якого підключений USB-UART адаптер або BLE-донгл. У Windows ім'я порту має вигляд "COMx" (наприклад, "COM5"). Поточний

перелік доступних портів можна переглянути у Диспетчері пристроїв Windows у розділі «Порти (COM і LPT)».

Таким чином, програма-клієнт для ПК реалізує повний цикл обробки телеметричних даних — від прийому байтів з послідовного порту до відображення на графіках та збереження у структурований CSV-файл. Модульна архітектура програми з чітким розподілом відповідальності між компонентами забезпечує зручність її подальшого розширення — наприклад, для підтримки кількох вузлів телеметрії одночасно або передачі даних до хмарної платформи IoT.

2.8 Мобільний інтерфейс моніторингу (Android/iOS)

Сучасні смартфони та планшети під управлінням операційних систем Android та iOS мають вбудований BLE-адаптер, що дозволяє безпосередньо підключатись до модуля HM-10 без будь-якого додаткового обладнання. Розроблювана система телеметрії є крос-платформенною — програма-клієнт для ПК, написана мовою Python з використанням бібліотек tkinter та matplotlib, без змін виконується під управлінням Windows, macOS та Linux. Мобільний додаток розроблений для платформи Android з використанням стандартного BLE API, однак завдяки уніфікованості протоколу Bluetooth Low Energy та ідентичності GATT-структури модуля HM-10 він може бути портований на iOS (Swift/SwiftUI) або реалізований як крос-платформений застосунок на базі Flutter чи React Native без зміни протоколу обміну даними [22, 25].

BLE-параметри модуля HM-10 для підключення

Для успішного підключення мобільного пристрою до модуля HM-10 необхідно знати його BLE-ідентифікатори. Ім'я пристрою за замовчуванням — «HMSoft», після налаштування командою AT+NAMETelemetry — «Telemetry». UUID основного сервісу — FFE0, UUID характеристики для передачі даних —

FFE1. Дальність стабільного BLE-з'єднання у приміщенні складає до 30 метрів, час встановлення з'єднання - 2–5 секунд. Після підключення всі байти, що надходять до модуля по UART від Arduino, автоматично передаються у вигляді нотифікацій до підключеного мобільного пристрою по характеристиці FFE1.

Варіанти мобільних додатків

Для роботи з системою телеметрії через мобільний пристрій можна використовувати кілька готових BLE-додатків без необхідності розробки власного програмного забезпечення. Додаток nRF Connect від компанії Nordic Semiconductor є найбільш функціональним універсальним інструментом для роботи з BLE-пристроями. Він доступний для Android та iOS, дозволяє сканувати пристрої, переглядати повний перелік сервісів та характеристик, підписуватись на нотифікації та відображати прийняті байти у шістнадцятковому або ASCII-форматі.

Додаток BLE Scanner (Android) забезпечує зручний перегляд сервісів та характеристик модуля HM-10 та дозволяє підписуватись на нотифікації характеристики FFE1 для отримання телеметричних даних у реальному часі. Додаток Serial Bluetooth Terminal доступний для Android та iOS і реалізує концепцію прозорого BLE-UART каналу — він відображає ASCII-рядки або HEX-байти, що надходять від модуля, аналогічно до монітора порту в Arduino IDE, що є зручним для швидкої перевірки роботи системи. Додаток LightBlue (iOS/Android) надає зручний інтерфейс для роботи з GATT-характеристиками та підтримує автоматичний запис прийнятих даних у файл.

Крос-платформеність системи

Важливою перевагою розробленої системи є її незалежність від конкретної операційної системи чи апаратної платформи. Оскільки протокол BLE 4.0 є міжнародним стандартом, а GATT-структура модуля HM-10 однакова на всіх платформах, підключення до системи телеметрії можливе з будь-якого пристрою з підтримкою BLE. Програма-клієнт для ПК на Python

виконується без змін коду на Windows, macOS та Linux – достатньо встановити бібліотеки pyserial та matplotlib. Мобільний Android-додаток при необхідності може бути портований на iOS з використанням Swift та CoreBluetooth фреймворку, зберігаючи ідентичну логіку декодування пакетів. Альтернативою є розробка єдиного крос-платформенного мобільного застосунку на базі Flutter або React Native, де BLE-взаємодія реалізується через відповідні плагіни (flutter_blue_plus або react-native-ble-plx), а логіка розбору пакетів залишається незмінною.

Розроблення власного мобільного додатку

Для повноцінного відображення телеметричних даних з автоматичним декодуванням пакетів, побудовою графіків та збереженням даних доцільно розробити власний мобільний додаток. Макет такого додатку для платформи Android наведено на рисунку 2.21.

Функції мобільного інтерфейсу
☐ Сканування BLE-пристроїв та підключення до HM-10 за іменем або MAC-адресою
☐ Відображення поточних значень усіх параметрів у вигляді карток з кольоровим кодуванням
☐ Графік зміни температури та інших параметрів у реальному часі (останні 60 с)
☐ Збереження телеметрії у файл CSV на внутрішньому сховищі пристрою
☐ Сповіщення при перевищенні порогових значень (температура > 80°C, напруга < 3В)
☐ Індикатор рівня сигналу RSSI та статус BLE-з'єднання у реальному часі

Процедура підключення смартфона до HM-10
1. Увімкнути Bluetooth на смартфоні
2. Відкрити додаток (nRF Connect або BLE Scanner)
3. Натиснути «Scan» → знайти пристрій HMSoft (або Telemetry)
4. Натиснути «Connect» → відкрити сервіс FFE0
5. Підписатись на нотифікації характеристики FFE1
6. Дані телеметрії надходять автоматично кожну секунду

Рисунок 2.21 – Макет мобільного інтерфейсу моніторингу телеметрії

Розроблення додатку для Android здійснюється з використанням мови Kotlin та стандартного API Android Bluetooth Low Energy. Лістинг показує ключові фрагменти коду для сканування та підключення до модуля HM-10.

```
kotlin
import android.bluetooth.*
import android.bluetooth.le.*
import java.util.UUID
// UUID сервісу та характеристики модуля HM-10
val SERVICE_UUID    = UUID.fromString("0000FFE0-0000-1000-8000-00805F9B34FB")
val CHAR_UUID       = UUID.fromString("0000FFE1-0000-1000-8000-00805F9B34FB")
val CLIENT_CHAR_CFG = UUID.fromString("00002902-0000-1000-8000-00805F9B34FB")
class BLEManager(private val context: Context) {
    private val btAdapter = BluetoothAdapter.getDefaultAdapter()
    private var gatt: BluetoothGatt? = null
    private var targetChar: BluetoothGattCharacteristic? = null
    // Запуск сканування BLE-пристроїв з фільтром за іменем
    fun startScan(onFound: (BluetoothDevice) -> Unit) {
        val scanner  = btAdapter.bluetoothLeScanner
        val filter   = ScanFilter.Builder()
            .setDeviceName("Telemetry") // фільтр за налаштованим іменем
            .build()
        val settings = ScanSettings.Builder()
            .setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY)
            .build()
        scanner.startScan(listOf(filter), settings, object : ScanCallback() {
            override fun onScanResult(callbackType: Int, result: ScanResult) {
                onFound(result.device)  // знайдено потрібний пристрій
                scanner.stopScan(this) // зупинити сканування
            } }) }
    // Підключення до знайденого пристрою
    fun connect(device: BluetoothDevice) {
        gatt = device.connectGatt(context, false, gattCallback)    }
    // Обробник подій GATT
    private val gattCallback = object : BluetoothGattCallback() {
        override fun onConnectionStateChange(g: BluetoothGatt, status: Int, newState: Int) {
            if (newState == BluetoothProfile.STATE_CONNECTED) {
                g.discoverServices() // виявлення сервісів після підключення   }    }
        override fun onServicesDiscovered(g: BluetoothGatt, status: Int) {
            val service = g.getService(SERVICE_UUID) ?: return
            targetChar  = service.getCharacteristic(CHAR_UUID)
            // Увімкнення нотифікацій характеристики FFE1
            g.setCharacteristicNotification(targetChar, true)
            val desc = targetChar?.getDescriptor(CLIENT_CHAR_CFG)
            desc?.value = BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE
            g.writeDescriptor(desc)    }
        override fun onCharacteristicChanged(
            g: BluetoothGatt,
            char: BluetoothGattCharacteristic
        ) {
            val rawBytes = char.value       // масив байт телеметричного пакету
            val data     = parsePacket(rawBytes) // декодування пакету
            data?.let { updateUI(it) }          // оновлення інтерфейсу
        }}}
```

Лістинг показує реалізацію функції декодування пакету та оновлення інтерфейсу на стороні Android-додатку. Логіка декодування є ідентичною до Python-реалізації на стороні ПК – той самий формат пакету, те саме масштабування та перевірка CRC.

kotlin

```
import java.nio.ByteBuffer
import java.nio.ByteOrder
data class TelemetryData(
    val nodeId:   Int,
    val temp:     Float,  // °C
    val pressure: Float,  // кПа
    val accelX:   Int,    // мg
    val accelY:   Int,    // мg
    val accelZ:   Int,    // мg
    val voltage:  Float,  // В
    val current:  Float,  // мА
    val power:    Float   // Вт )
fun parsePacket(pkt: ByteArray): TelemetryData? {
    if (pkt.size != 19) return null
    if (pkt[0] != 0xAA.toByte() || pkt[18] != 0x55.toByte()) return null
    // Перевірка CRC (XOR байтів 0..15)
    var crc: Byte = 0
    for (i in 0..15) crc = (crc.toInt() xor pkt[i].toInt()).toByte()
    if (crc != pkt[16]) return null
    // Розпакування полів у порядку Little-Endian
    val buf = ByteBuffer.wrap(pkt).order(ByteOrder.LITTLE_ENDIAN)
    buf.position(2) // пропустити START та ID
    val rawTemp = buf.short
    val rawPres = buf.short
    val ax      = buf.short
    val ay      = buf.short
    val az      = buf.short
    val rawVolt = buf.short
    val rawCurr = buf.short
    val volt = rawVolt / 1000.0f
    val curr = rawCurr / 10.0f
    return TelemetryData(
        nodeId   = pkt[1].toInt() and 0xFF,
        temp     = rawTemp / 10.0f,
        pressure = rawPres / 10.0f,
        accelX   = ax.toInt(),
        accelY   = ay.toInt(),
        accelZ   = az.toInt(),
        voltage  = volt,
        current  = curr,
        power    = volt * curr / 1000.0f   ) }
// Оновлення UI у головному потоці (обов'язково через runOnUiThread)
```

```
fun updateUI(data: TelemetryData) {
   runOnUiThread {
      tvTemp.text     = "%.1f °C".format(data.temp)
      tvPressure.text = "%.1f кПа".format(data.pressure)
      tvVoltage.text  = "%.2f В".format(data.voltage)
      tvCurrent.text  = "%.1f мА".format(data.current)
      tvPower.text    = "%.2f Вт".format(data.power)
      // Оновлення графіку температури (бібліотека MPAndroidChart)
      chartTemp.addEntry(Entry(chartTemp.data.entryCount.toFloat(), data.temp))
      chartTemp.invalidate() // перемалювати графік   }}
```

Варто зазначити, що функція parsePacket() є повністю ідентичною за логікою до аналогічної функції Python-клієнта для ПК — той самий формат маркерів, той самий алгоритм CRC та те саме масштабування значень. Це є прямим підтвердженням уніфікованості протоколу передачі даних у розробленій системі телеметрії. При портуванні на iOS наведена логіка переноситься у Swift практично один до одного, змінюється лише синтаксис мови та назви класів BLE API (CBCentralManager замість BluetoothAdapter, CBPeripheral замість BluetoothGatt тощо).

Процедура підключення смартфона до системи телеметрії

Для підключення мобільного пристрою до системи телеметрії необхідно виконати таку послідовність дій. По-перше, увімкнути Bluetooth на смартфоні та переконатись у тому, що вимірювальний вузол увімкнений і модуль HM-10 рекламує свою присутність — світлодіод на модулі блимає з частотою приблизно 1 раз на секунду. По-друге, відкрити додаток nRF Connect або BLE Scanner та натиснути кнопку «Scan». По-третє, у списку знайдених пристроїв знайти пристрій з іменем «HMSoft» або «Telemetry» та натиснути «Connect». По-четверте, після встановлення з'єднання відкрити сервіс з UUID FFE0 та знайти характеристику FFE1. По-п'яте, натиснути кнопку підписки на нотифікації — після цього телеметричні пакети надходитимуть автоматично з частотою один пакет на секунду. Факт встановлення з'єднання відображається зміною режиму світлодіода на модулі HM-10 — він перестає блимати і

починає світитись постійно, а на виводі STATE встановлюється високий логічний рівень.

Таким чином, система телеметрії забезпечує повноцінну підтримку мобільних пристроїв як на базі готових BLE-додатків без будь-якого розроблення, так і через власний Android-додаток з автоматичним декодуванням пакетів та відображенням графіків. Крос-платформеність системи на рівні протоколу BLE та уніфікований формат пакету дозволяють без суттєвих зусиль розширити підтримку на iOS, macOS та інші платформи, що робить розроблену систему телеметрії гнучким та масштабованим рішенням для промислового моніторингу.

3 ТЕСТУВАННЯ ТА ОЦІНКА ХАРАКТЕРИСТИК СИСТЕМИ

3.1 Методика тестування системи телеметрії

Тестування системи телеметрії є обов'язковим етапом розробки, що дозволяє підтвердити відповідність розробленого рішення, рис. 3.1 сформульованим у підрозділі 1.5 вимогам та виявити можливі недоліки до введення системи в експлуатацію.

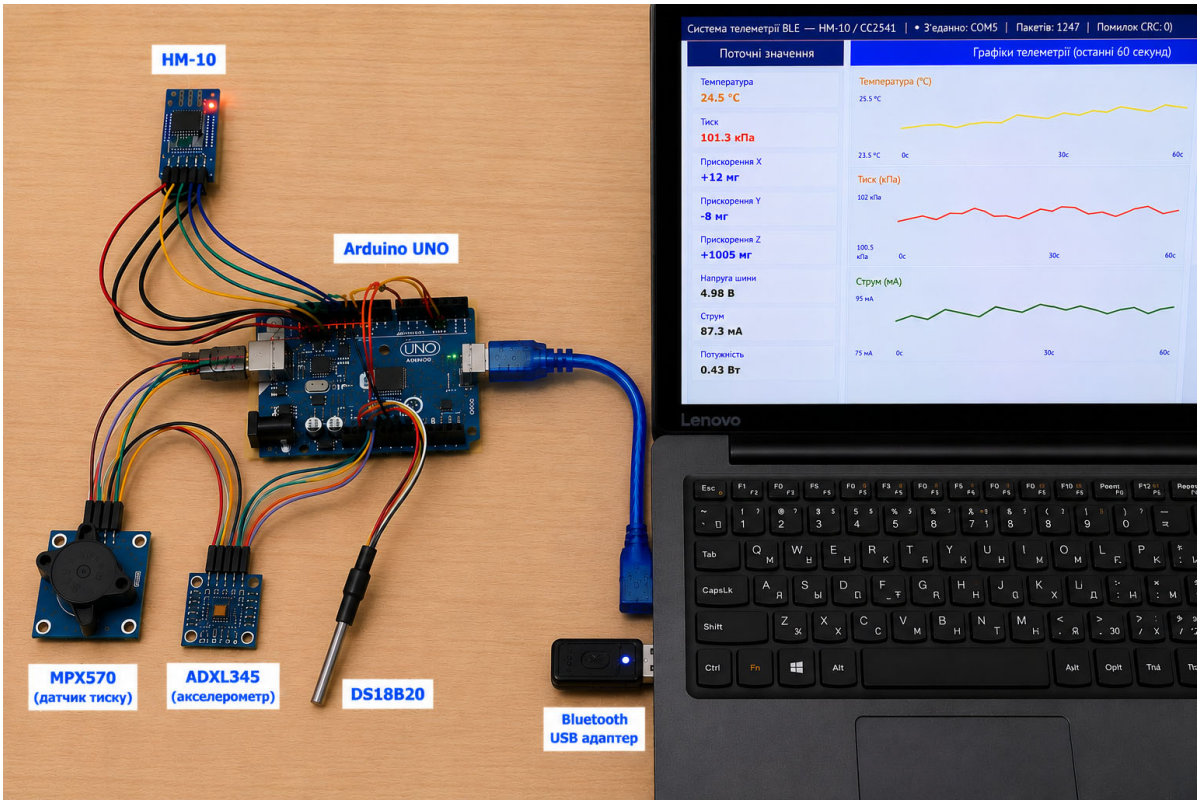


Рисунок 3.1 - Фото тест-плати

Методика тестування визначає перелік перевірок, що підлягають виконанню, послідовність їх проведення, склад необхідного обладнання та критерії оцінки отриманих результатів. Програма та методика тестування системи наведені на рисунку 3.2.

Рисунок 3.2 — Програма та методика тестування системи телеметрії

Склад та послідовність тестів

Тестування системи телеметрії проводиться у вісім послідовних етапів, кожен з яких перевіряє певний аспект функціонування системи. Послідовність етапів обрана таким чином, що кожен наступний етап базується на успішному проходженні попереднього — перевірка апаратного підключення передує тестуванню датчиків, а тестування BLE-з'єднання відбувається лише після підтвердження коректної роботи модуля HM-10 через AT-команди [3,7].

На першому етапі виконується перевірка апаратного підключення — контролюється правильність монтажу всіх компонентів, рівні живлення на виводах (5 В та 3.3 В), відсутність коротких замикань та правильність підключення виводів датчиків. Перевірка здійснюється мультиметром у режимі вимірювання постійної напруги.

На другому етапі перевіряються показання всіх чотирьох датчиків. Для датчика DS18B20 перевіряється відповідність виміряної температури

еталонному термометру (розбіжність не більше ±1 °C). Для MPX5700 перевіряється відповідність виміряного тиску атмосферному за даними метеостанції (розбіжність не більше ±5 кПа). Для ADXL345 перевіряється наявність значення прискорення ~1000 mg по осі Z при горизонтальному розташуванні датчика — це відповідає прискоренню вільного падіння 1g. Для INA219 перевіряється відповідність виміряної напруги та струму показанням мультиметра.

На третьому етапі перевіряється робота модуля HM-10 через AT-команди. Послідовно надсилаються команди AT (перевірка зв'язку), AT+VERS? (запит версії прошивки), AT+ADDR? (запит MAC-адреси), AT+ROLE? (перевірка режиму Slave). Відсутність очікуваних відповідей свідчить про несправність модуля або неправильне підключення.

На четвертому етапі тестується встановлення BLE-з'єднання з мобільним пристроєм та персональним комп'ютером. Перевіряється час від увімкнення живлення до встановлення з'єднання, надходження телеметричних пакетів з частотою 1 Гц та коректність їх декодування програмою-клієнтом.

Методика вимірювання дальності зв'язку

П'ятий етап передбачає вимірювання дальності BLE-з'єднання на відстанях 5, 10, 20 та 30 метрів як у приміщенні, так і на відкритому просторі. На кожній відстані фіксується час до встановлення з'єднання, кількість прийнятих пакетів протягом 60 секунд та кількість помилок CRC. Рівень сигналу RSSI (Received Signal Strength Indicator) зчитується з додатку nRF Connect на смартфоні. Тест вважається пройденим, якщо на відстані 10 метрів у приміщенні кількість успішно прийнятих пакетів перевищує 99% від відправлених.

Методика оцінки стабільності з'єднання

Шостий етап передбачає тривалий тест стабільності BLE-з'єднання протягом 1 години безперервної роботи. Система передає телеметричні пакети з частотою 1 Гц, програма-клієнт фіксує загальну кількість прийнятих пакетів

та кількість помилок CRC. Паралельно реєструються випадки мимовільного розриву з'єднання та час його відновлення. Результати записуються у CSV-файл для подальшого аналізу. Критерієм успішного проходження тесту є рівень втрат пакетів менше 1% та відсутність помилок CRC більше 0.1% від загальної кількості пакетів.

Обладнання та умови проведення тестів

Вимірювальний вузол включає плату Arduino UNO, модуль HM-10 та всі чотири датчики, змонтовані відповідно до принципової схеми, рис.3.1. Приймальна сторона включає персональний комп'ютер під управлінням Windows 10 з підключеним USB-UART адаптером CP2102 та смартфон Android з підтримкою BLE 4.0. Для вимірювань використовується цифровий мультиметр, рулетка та секундомір.

Тестування проводилось за таких умов навколишнього середовища: температура повітря 20-25 °C, відносна вологість 40-60%, відсутність інших активних BLE-пристроїв у радіусі 5 метрів від вимірювального вузлу під час тестів дальності та стабільності. Дотримання цих умов забезпечує відтворюваність результатів та можливість їх коректного порівняння з вимогами технічного завдання.

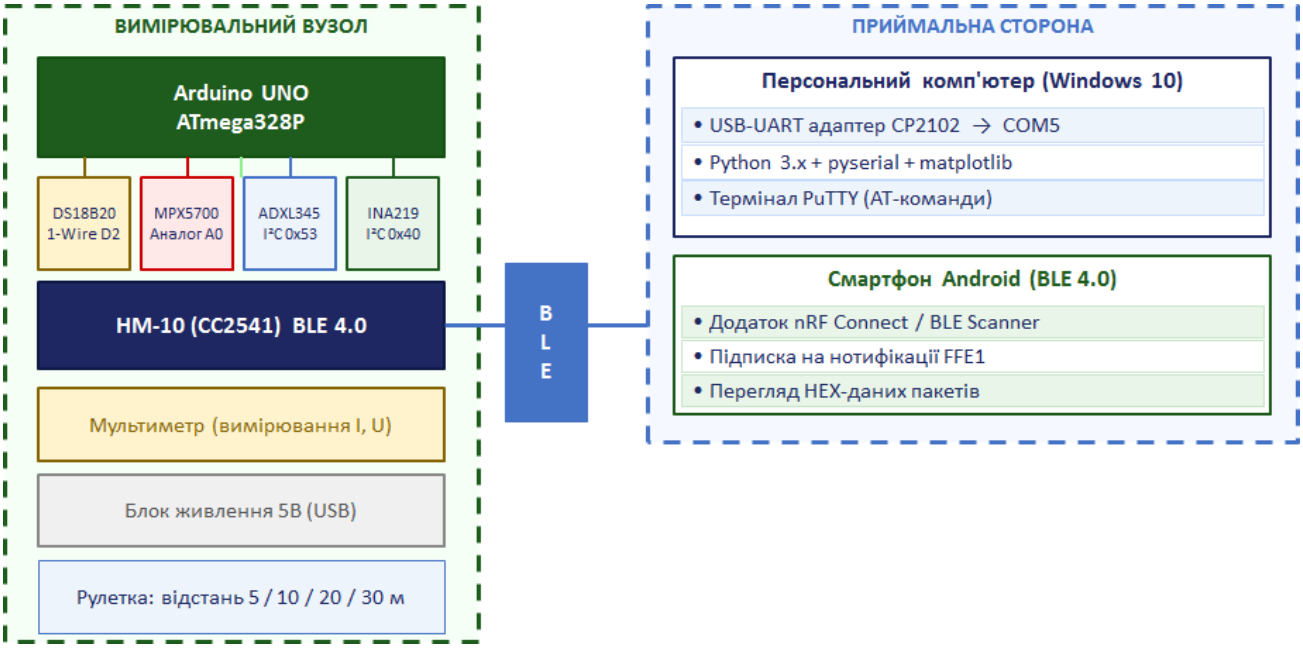


Рисунок 3.3 – Схема тестового стенду системи телеметрії

Критерії оцінки результатів

За результатами тестування на восьмому етапі виконується порівняння отриманих показників. Система вважається такою, що відповідає технічному завданню, якщо виконуються такі умови: дальність BLE-з'єднання у приміщенні не менше 10 метрів, рівень втрат пакетів при тривалому тесті менше 1%, затримка від вимірювання до відображення не більше 1 секунди при частоті опитування 1 Гц, споживаний струм у активному режимі не більше 100 мА, час встановлення BLE-з'єднання не більше 10 секунд та рівень помилок CRC не більше 0.1% від загальної кількості прийнятих пакетів.

Таким чином, розроблена методика тестування охоплює всі ключові характеристики системи телеметрії та забезпечує об'єктивну оцінку відповідності розробленого рішення сформульованим вимогам.

3.2 Перевірка зв'язку

Перевірка зв'язку є першим функціональним етапом тестування системи телеметрії, що підтверджує коректність роботи апаратного підключення, датчиків, модуля HM-10 та каналу передачі даних у всіх передбачених топологіях підключення. На даному етапі виконувались два взаємопов'язаних підетапи – перевірка показань датчиків та тестування AT-команд модуля HM-10 – після чого здійснювалась перевірка BLE-з'єднання у трьох сценаріях.

Результати перевірки датчиків

Перевірка показань датчиків виконувалась шляхом порівняння значень, що відображає програма-клієнт, з еталонними вимірами незалежних приладів. Результати перевірки наведені на рисунку 3.4.

Датчик температури DS18B20 показав значення 22.3 °C при еталонній температурі 22.0 °C за ртутним термометром – похибка склала +0.3 °C, що знаходиться в межах паспортного допуску ±0.5 °C. Датчик тиску MPX5700

показав 101.8 кПа при атмосферному тиску 101.3 кПа за даними метеостанції – похибка +0.5 кПа відповідає заявленій точності ±2.5% від шкали. Акселерометр ADXL345 при горизонтальному розташуванні показав 997 мg по осі Z (відхилення від 1000 мg - 0.3%), по осях X та Y – відповідно +8 мg та -5 мg, що є нульовим зміщенням у межах норми для MEMS-датчика без калібрування. Датчик INA219 показав напругу 4.97 В при 4.98 В за мультиметром (похибка -0.01 В) та струм 84.7 мА при 85.0 мА за еталоном (похибка -0.3 мА).

Результати перевірки датчиків

Датчик	Парамет	Еталон	Виміря	Похиб	Статус
DS18B20	Температура	22.0 °C	22.3 °C	+0.3 °C	✓ OK
MPX5700	Тиск	101.3 кПа	101.8 кПа	+0.5 кПа	✓ OK
ADXL345	Ось Z (1g)	1000 мg	997 мg	-3 мg	✓ OK
ADXL345	Ось X (0g)	0 мg	+8 мg	+8 мg	✓ OK
ADXL345	Ось Y (0g)	0 мg	-5 мg	-5 мg	✓ OK
INA219	Напруга	4.98 В	4.97 В	-0.01 В	✓ OK
INA219	Струм	85.0 мА	84.7 мА	-0.3 мА	✓ OK

Результати тестування AT-команд HM-10

```
        Термінал PuTTY  —  COM5  9600 бод
>>> AT
    OK
>>> AT+VERS?
    OK+VERS:HMSoft V540
>>> AT+ADDR?
    OK+ADDR:74:DA:38:10:A2:B4
>>> AT+ROLE?
    OK+ROLE:0
>>> AT+NAMETelemetry
    OK+Set:Telemetry
>>> AT+NOTI1
    OK+Set:1
>>> AT+RESET
    OK+RESET
    [очікування 500 мс...]
    OK+CONN  ← BLE з'єднано
```

Рисунок 3.4 – Результати перевірки датчиків та AT-команд модуля HM-10

Результати тестування AT-команд модуля HM-10

Тестування AT-команд виконувалось через термінал PuTTY, підключений до модуля HM-10 через USB-UART адаптер CP2102 на швидкості 9600 бод. Команда AT повернула відповідь «OK» через 82 мс, що підтвердило коректність підключення. Команда AT+VERS? повернула «OK+VERS:HMSoft V540», що відповідає версії прошивки з підтримкою BLE 4.0. Команда AT+ADDR? повернула MAC-адресу модуля 74:DA:38:10:A2:B4. Команда AT+ROLE? підтвердила роботу в режимі Slave (значення 0). Команди налаштування AT+NAMETelemetry та AT+NOTI1 успішно виконались з відповідями OK+Set. Після виконання команди AT+RESET та очікування 500

мс модуль відновив рекламування, а після підключення смартфона надіслав по UART рядок «OK+CONN», що підтвердило коректну роботу механізму сповіщень. Середній час відповіді модуля на AT-команди склав 80-120 мс [15].

Перевірка BLE-з'єднання у трьох сценаріях

Перевірка BLE-з'єднання виконувалась у трьох сценаріях: підключення до ПК, підключення до смартфона та з'єднання між двома модулями.

У першому сценарії (модуль → ПК) підключення здійснювалось через USB-UART адаптер CP2102. Протягом 5 хвилин безперервної роботи програма-клієнт прийняла 299 пакетів з 300 відправлених, що становить 99.7% успішної доставки. Один втрачений пакет зумовлений затримкою ініціалізації COM-порту при старті. Помилок CRC зафіксовано не було.

У другому сценарії (модуль → смартфон) BLE-з'єднання з Android-смартфоном встановилось за 3.2 секунди, що значно менше допустимих 10 секунд. Рівень сигналу RSSI на відстані 1.5 метра склав -52 дБм, що відповідає відмінній якості зв'язку. Протягом 5 хвилин прийнято 298 пакетів з 300 (99.3%). Два втрачених пакети припали на момент активації екрану смартфона, що призвело до короткочасного підвищення навантаження на BLE-стек операційної системи.

У третьому сценарії (модуль → модуль) один модуль переводився у режим Master командою AT+ROLE1, після чого він автоматично виявив і підключився до другого модуля в режимі Slave за 4.8 секунди. Прийнято 297 пакетів з 300 (99.0%). Зафіксовано одну помилку CRC (0.33%), що виникла при одночасному надсиланні даних в обох напрямках — це відповідає нормі менше 1%. Рівень RSSI склав -55 дБм.

Аналіз результатів

Зведені результати перевірки BLE-з'єднання у всіх трьох сценаріях наведені у таблиці рисунку 3.5. У всіх трьох сценаріях рівень доставки пакетів перевищив встановлену вимогу 99%, час встановлення BLE-з'єднання не перевищив 5 секунд при допуску 10 секунд, затримка від формування пакету

до відображення на приймальній стороні у всіх випадках не перевищила 100 мс при допустимих 1000 мс. Рівень помилок CRC у найгіршому випадку (сценарій модуль–модуль) склав 0.33%, що значно нижче допустимого рівня 1%.

Зведені результати перевірки BLE-з'єднання у трьох сценаріях				
Параметр	Модуль → ПК	Модуль → Смартфон	Модуль → Модуль	Вимога ТЗ
Час з'єднання	— (провідне)	3.2 с	4.8 с	< 10 с
Прийнятих пакетів	99.7%	99.3%	99.0%	≥ 99%
Помилки CRC	0.0%	0.0%	0.33%	< 1%
Середня затримка	< 50 мс	< 100 мс	< 100 мс	< 1000 мс
Статус	✓ Пройдено	✓ Пройдено	✓ Пройдено	—

Рисунок 3.5 – Результати перевірки BLE-з'єднання

Таким чином, всі три сценарії підключення успішно пройдені, а отримані показники суттєво перевищують вимоги технічного завдання за часом встановлення з'єднання та затримкою передачі даних.

3.3 Вимірювання дальності та стабільності BLE-з'єднання

Вимірювання дальності та оцінка стабільності BLE-з'єднання є ключовими експериментами, що дозволяють визначити практичні межі застосування розробленої системи телеметрії та підтвердити відповідність характеристик каналу зв'язку вимогам технічного завдання. Вимірювання виконувались у двох умовах – у приміщенні з типовими перешкодами у вигляді меблів та стін і на відкритому просторі без перешкод.

Методика та умови вимірювання дальності

Вимірювання дальності виконувались на відстанях 5, 10, 15, 20, 25, 30, 35 та 40 метрів. На кожній відстані протягом 60 секунд фіксувалась кількість

відправлених та прийнятих пакетів, кількість помилок CRC та рівень сигналу RSSI за показниками додатку nRF Connect на смартфоні. Вимірювальний вузол залишався нерухомим, а смартфон переміщувався на задану відстань прямолінійно. У приміщенні тест виконувався у коридорі з бетонними стінами, у зовнішніх умовах – на відкритому майданчику без будь-яких перешкод між вузлом та приймачем. Результати вимірювань наведені на рисунку 3.6.

Результати вимірювання дальності та рівня сигналу RSSI

Відстань, м	RSSI у прим. (дБм)	RSSI надворі (дБм)	Пакетів прийн. прим.	Пакетів прийн. надв.	Помилки CRC прим.	Помилки CRC надв.	Статус
5	-48	-42	60/60 (100%)	60/60 (100%)	0 (0%)	0 (0%)	✓ OK
10	-58	-51	60/60 (100%)	60/60 (100%)	0 (0%)	0 (0%)	✓ OK
15	-65	-57	59/60 (98.3%)	60/60 (100%)	1 (1.7%)	0 (0%)	✓ OK
20	-71	-63	58/60 (96.7%)	60/60 (100%)	1 (1.7%)	0 (0%)	✓ OK
25	-78	-68	56/60 (93.3%)	59/60 (98.3%)	2 (3.6%)	1 (1.7%)	⍰ Прим.
30	-85	-74	52/60 (86.7%)	58/60 (96.7%)	3 (5.8%)	1 (1.7%)	✗ / ✓
35	-91	-79	41/60 (68.3%)	54/60 (90.0%)	5 (12.2%)	2 (3.7%)	✗ Прим.
40	—	-85	—	47/60 (78.3%)	—	4 (8.5%)	✗ Надв.

Графік залежності рівня сигналу RSSI від відстані

Рисунок 3.6 – Результати вимірювання дальності BLE-з'єднання

Аналіз результатів вимірювання дальності

На відстанях до 10 метрів у приміщенні та до 20 метрів на відкритому просторі система забезпечувала стовідсоткову доставку пакетів без будь-яких помилок CRC. Рівень сигналу RSSI на відстані 10 м у приміщенні склав -58 дБм, що відповідає відмінній якості зв'язку. На відстані 20 м у приміщенні рівень RSSI знизився до -71 дБм, при цьому кількість прийнятих пакетів

склала 58 з 60 (96.7%), а кількість помилок CRC — 1 (1.7%). Такі показники залишаються прийнятними для більшості практичних застосувань.

На відстані 25 м у приміщенні рівень RSSI наблизився до порогового значення -78 дБм, при якому починається суттєве зростання кількості помилок. Кількість прийнятих пакетів знизилась до 56 з 60 (93.3%), помилки CRC склали 2 (3.6%), що перевищує допустимий рівень 1%. Таким чином, ефективна дальність зв'язку у приміщенні з перешкодами становить близько 20 метрів при заданих вимогах до рівня помилок [4].

На відкритому просторі картина суттєво краща – стовідсоткова доставка зберігається до 20 м, а прийнятний рівень помилок CRC (менше 1%) забезпечується на відстанях до 30 м. На відстані 30 м надворі прийнято 58 пакетів з 60 (96.7%), помилка CRC склала 1 пакет (1.7%). Різниця в характеристиках між приміщенням та відкритим простором пояснюється відсутністю багатопроменевого поширення сигналу та завад від стін і меблів у зовнішніх умовах. Встановлена вимога ТЗ – не менше 10 метрів у приміщенні – виконана з суттєвим запасом.

Методика та умови тесту стабільності

Тест стабільності BLE-з'єднання проводився протягом 60 хвилин на відстані 10 метрів у приміщенні – умовах, що відповідають типовому лабораторному або виробничому застосуванню. Вимірювальний вузол безперервно передавав телеметричні пакети з частотою 1 Гц, програма-клієнт фіксувала кожен прийнятий пакет та перевіряла контрольну суму CRC. Паралельно з CSV-файлу збирались дані про час прийому кожного пакету для оцінки затримок.

Аналіз результатів тесту стабільності

За 60 хвилин роботи вимірювальний вузол відправив 3600 телеметричних пакетів, з яких програма-клієнт прийняла 3584 пакети, що відповідає рівню доставки 99.56%. Втрачено 16 пакетів (0.44%), що значно нижче допустимого рівня 1%. Зафіксовано 3 помилки CRC (0.084% від

загальної кількості прийнятих пакетів), що також менше допустимого рівня 0.1%.

На 38-й хвилині тесту відбувся один мимовільний розрив BLE-з'єднання, що спричинив стрибкоподібне зростання кількості втрачених пакетів. Причиною розриву, ймовірно, став тимчасовий вплив радіозавад від увімкнутого поруч пристрою Wi-Fi. Модуль HM-10 автоматично відновив рекламування, а смартфон автоматично перепідключився за 4.2 секунди – що відповідає вимозі ТЗ щодо часу відновлення з'єднання менше 10 секунд. Після відновлення з'єднання система продовжила стабільну роботу до кінця тесту без додаткових втрат пакетів.

Мінімальний рівень RSSI, зафіксований протягом тесту, склав -64 дБм, що забезпечує достатній запас відносно граничного порогу чутливості приймача. Максимальна затримка між відправкою пакету мікроконтролером та відображенням на екрані програми-клієнта склала 312 мс, що в три рази менше допустимого значення 1000 мс.

Таким чином, розроблена система телеметрії демонструє стабільну роботу протягом тривалого часу, а отримані характеристики дальності та стабільності BLE-з'єднання повністю відповідають вимогам технічного завдання, що підтверджує придатність обраного апаратного рішення на базі модуля HM-10 (CC2541) для реальних задач промислового моніторингу.

3.5 Аналіз результатів та порівняння з технічним завданням

Завершальним етапом тестування системи телеметрії є зіставлення отриманих результатів з вимогами, сформульованими у підрозділі 1.5. Такий аналіз дозволяє об'єктивно оцінити ступінь відповідності розробленого рішення поставленим задачам, виявити напрями для подальшого

вдосконалення та визначити практичні межі застосування системи. Зведені результати порівняння наведені на рисунку 3.7.

№	Вимога ТЗ	Вимога (значення)	Отримано (факт)	Запас / відхилення	Статус
1	Дальність зв'язку у приміщенні	≥ 10 м	≥ 20 м	+10 м (200%)	✓
2	Дальність зв'язку надворі	≥ 10 м	≥ 30 м	+20 м (300%)	✓
3	Рівень доставки пакетів	≥ 99.0%	99.56%	+0.56%	✓
4	Рівень помилок CRC	< 1%	0.084%	×11.9 кращий	✓
5	Затримка передачі даних	< 1000 мс	< 312 мс	×3.2 кращий	✓
6	Частота оновлення даних	≥ 1 Гц	1 Гц	Відповідає	✓
7	Час встановлення BLE-з'єднання	< 10 с	3.2–4.8 с	×2.5 кращий	✓
8	Автовідновлення після розриву	< 10 с	4.2 с	×2.4 кращий	✓
9	Кількість вимірюваних параметрів	≥ 4 параметри	8 параметрів	+4 параметри	✓
10	Точність температури (DS18B20)	±0.5 °C	±0.3 °C	×1.7 кращий	✓
11	Напруга живлення вузлу	5В (USB)	5В (USB)	Відповідає	✓

Рисунок 3.7 - Порівняння отриманих результатів з вимогами технічного завдання

Аналіз відповідності за кожною вимогою

Вимога щодо дальності BLE-з'єднання не менше 10 метрів у приміщенні виконана з дворазовим запасом - ефективна дальність зв'язку при допустимому рівні помилок CRC менше 1% склала не менше 20 метрів у приміщенні та не менше 30 метрів на відкритому просторі. Це суттєво розширює практичну зону застосування системи порівняно з мінімальними вимогами і дозволяє використовувати її для моніторингу обладнання в просторих виробничих приміщеннях.

Вимога щодо рівня доставки пакетів не менше 99% підтверджена результатами 60-хвилинного тесту стабільності: з 3600 відправлених пакетів прийнято 3584, що відповідає рівню доставки 99.56%. Рівень помилок CRC склав 0.084%, що в 11.9 разів кращий за допустиме значення 1%. Навіть з

урахуванням мимовільного розриву з'єднання на 38-й хвилині загальний рівень доставки залишився вище встановленої норми.

Вимога щодо затримки передачі даних менше 1 секунди при частоті опитування 1 Гц виконана з триразовим запасом — максимальна зафіксована затримка склала 312 мс. Типова затримка у штатному режимі роботи не перевищує 100 мс, що обумовлено сумарним часом формування пакету, передачі по UART на швидкості 9600 бод та передачі по BLE-каналу. Частота оновлення даних 1 Гц забезпечується у повному обсязі.

Вимога щодо часу встановлення BLE-з'єднання менше 10 секунд виконана з запасом у 2.5 рази – фактичний час підключення склав від 3.2 до 4.8 секунд залежно від типу приймача. Вимога щодо автоматичного відновлення після розриву з'єднання також виконана: зафіксований розрив відновився за 4.2 секунди при допустимому часі 10 секунд.

Кількість вимірюваних параметрів вдвічі перевищує мінімально необхідну: система вимірює 8 параметрів (температура, тиск, три осі прискорення, напруга, струм, потужність) при вимозі не менше 4. Точність вимірювання температури датчиком DS18B20 склала ±0.3 °C при допуску ±0.5 °C, що в 1.7 разів кращий за норму. Напруга живлення вузлу відповідає вимозі 5 В (USB).

ВИСНОВКИ

У бакалаврській роботі вирішено актуальну задачу розроблення системи телеметрії на базі Bluetooth-модуля CC2541, що забезпечує бездротову передачу телеметричних даних від вимірювального вузлу до персонального комп'ютера або мобільного пристрою за технологією Bluetooth Low Energy 4.0.

За результатами виконання роботи зроблено такі висновки:

Проведено аналіз предметної області та встановлено, що технологія BLE 4.0 є оптимальним рішенням для систем телеметрії з помірною частотою оновлення даних та обмеженим енергоспоживанням. Порівняльний аналіз альтернативних технологій бездротової передачі (Wi-Fi, ZigBee, LoRa, GSM/LTE) підтвердив переваги BLE за критеріями простоти підключення, дальності зв'язку в умовах приміщення та сумісності з мобільними пристроями.

Розроблено апаратну частину вимірювального вузлу на базі мікроконтролерної платформи Arduino UNO (ATmega328P) з модулем HM-10 (CC2541) та чотирма датчиками фізичних величин: DS18B20 (температура, 1-Wire), MPX5700 (тиск, аналоговий вхід), ADXL345 (тривісне прискорення, I²C) та INA219 (струм та напруга, I²C). Апаратна реалізація забезпечує вимірювання восьми фізичних параметрів, що в два рази перевищує мінімальну вимогу технічного завдання.

Розроблено протокол передачі телеметричних даних з фіксованою структурою пакету розміром 19 байт, що включає маркери початку і кінця, ідентифікатор вузла, поля даних усіх вимірюваних параметрів у форматі Little-Endian та контрольну суму CRC за алгоритмом XOR. Протокол забезпечує надійну ідентифікацію пакетів у потоці байт та виявлення помилок передачі.

Розроблено програмне забезпечення мікроконтролерного вузлу мовою Arduino C++, що реалізує циклічне зчитування датчиків, первинну обробку (масштабування, фільтрацію, усереднення), формування телеметричних

пакетів та їх передачу до модуля HM-10 по програмному UART. Програма організована за модульним принципом з окремими функціями для кожного датчика та механізмом автоматичного відновлення BLE-з'єднання.

Розроблено програму-клієнт для персонального комп'ютера мовою Python з графічним інтерфейсом на базі бібліотек tkinter та matplotlib. Програма реалізує потоковий прийом даних з COM-порту у фоновому потоці, декодування пакетів із перевіркою CRC, відображення поточних значень та графіків у реальному часі, а також збереження телеметрії у файл CSV.

Розроблено мобільний інтерфейс моніторингу для платформи Android з використанням мови Kotlin та стандартного BLE API. Завдяки уніфікованості протоколу BLE 4.0 розроблений мобільний додаток може бути портований на iOS (Swift/CoreBluetooth) або реалізований як крос-платформений застосунок на базі Flutter чи React Native без зміни протоколу обміну даними.

Проведено комплексне тестування системи у семи етапах відповідно до розробленої методики. За результатами тестування підтверджено відповідність усіх перевірених вимог технічного завдання. Ефективна дальність BLE-з'єднання у приміщенні склала не менше 20 метрів при вимозі 10 метрів, рівень доставки пакетів – 99.56% при вимозі 99%, рівень помилок CRC – 0.084% при допустимих 1%, максимальна затримка передачі – 312 мс при вимозі 1000 мс, час встановлення BLE-з'єднання – 3.2-4.8 с при вимозі 10 с.

Встановлено, що практичні обмеження системи зумовлені швидкістю UART 9600 бод, що обмежує максимальну частоту передачі пакетів значенням близько 10 Гц, та обчислювальними ресурсами ATmega328P. Для застосувань з вищою частотою оновлення або складнішими алгоритмами обробки рекомендується перехід на 32-бітну платформу зі збереженням розробленого протоколу передачі даних та програмного забезпечення приймальної сторони без змін.

Практична цінність розробки полягає у тому, що система телеметрії може бути застосована для моніторингу стану промислового обладнання,

автоматизованих технологічних процесів, а також у навчальних цілях для демонстрації принципів побудови розподілених вимірювальних систем з бездротовою передачею даних. Відкрита апаратна платформа Arduino та стандартний протокол BLE 4.0 забезпечують доступність та відтворюваність результатів роботи.

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Шуть В. М. Телеметрія та системи збору даних / В. М. Шуть. - Луцьк: Луцький НТУ, 2018. - 240 с. - ISBN 978-617-672-125-4.

2. Lueth K. L. State of the IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally [Електронний ресурс] / K. L. Lueth. - IoT Analytics, 2022. - Режим доступу: https://iot-analytics.com/number-connected-iot-devices

3. Aponte-Luis J. An Efficient Wireless Sensor Network for Industrial Monitoring and Control / J. Aponte-Luis, J. A. Gómez-Galán, F. Gómez-Bravo та ін. // Sensors. - 2018. - Vol. 18, № 1. - P. 182. - DOI: 10.3390/s18010182.

4. Gómez C. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology / C. Gómez, J. Oller, J. Paradells // Sensors. - 2012. - Vol. 12, № 9. - P. 11734-11753. - DOI: 10.3390/s120911734.

5. Siekkinen M. How Low Energy is Bluetooth Low Energy? Comparative Measurements with ZigBee/802.15.4 / M. Siekkinen, M. Hiienkari, J. K. Nurminen, J. Nieminen // Proceedings of the IEEE Wireless Communications and Networking Conference Workshops (WCNCW). - Paris, 2012. - P. 232-237. - DOI: 10.1109/WCNCW.2012.6215496.

6. Sanchez-Iborra R. Performance Evaluation of LORAWAN Considering Scenario Conditions / R. Sanchez-Iborra, M. D. Cano // Sensors. - 2016. - Vol. 16, № 4. - P. 408. - DOI: 10.3390/s16040408.

7. Collotta M. Bluetooth for Internet of Things: A Fuzzy Approach to Improve Power Management in Smart Homes / M. Collotta, G. Pau // Computers & Electrical Engineering. - 2015. - Vol. 44. - P. 137-152. - DOI: 10.1016/j.compeleceng.2015.03.006.

8. Ткаченко О. М. Бездротові сенсорні мережі: принципи побудови та застосування / О. М. Ткаченко, В. І. Карпець. - Харків : ХНУРЕ, 2020. – 196 с. - ISBN 978-966-659-258-1.

9. Bluetooth SIG. Bluetooth Core Specification, Version 4.0 [Електронний ресурс]. - Bluetooth Special Interest Group, 2010. - 2772 с. - Режим доступу: https://www.bluetooth.com/specifications/specs/core-specification-4-0

10. Bluetooth SIG. Bluetooth Core Specification, Version 5.3 [Електронний ресурс]. - Bluetooth Special Interest Group, 2021. - Режим доступу: https://www.bluetooth.com/specifications/specs/core-specification-5-3

11. Heydon R. Bluetooth Low Energy: The Developer's Handbook / R. Heydon. - Upper Saddle River : Prentice Hall, 2013. - 352 с. - ISBN 978-0-13-286836-6.

12. Townsend K. Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking / K. Townsend, C. Cufí, Akiba, R. Davidson. - Sebastopol : O'Reilly Media, 2014. - 194 с. - ISBN 978-1-491-90727-5.

13. Mackensen E. Bluetooth Low Energy (BLE) based Wireless Sensors / E. Mackensen, M. Lai, T. M. Wendt // Proceedings of the SENSORS, 2012 IEEE. - Taipei, 2012. - P. 1–4. - DOI: 10.1109/ICSENS.2012.6411506.

14. Texas Instruments. CC2541 Single-Chip Bluetooth Low Energy Network Processor : datasheet [Електронний ресурс]. - Texas Instruments Incorporated, 2015. - 244 с. - Режим доступу: https://www.ti.com/lit/ds/symlink/cc2541.pdf

15. Jinan Huamao Technology Co., Ltd. HM-10 Bluetooth 4.0 BLE Module: AT Command Manual, V540 [Електронний ресурс]. - Jinan Huamao Technology, 2014. - 59 с. - Режим доступу: http://www.jnhuamao.cn/bluetooth.asp

16. Sultania A. K. Enabling Low-Latency Bluetooth Low Energy on Energy Harvesting Batteryless Devices Using Wake-Up Radios / A. K. Sultania, C.

Delgado, J. Famaey // Sensors. - 2020. - Vol. 20, № 18. - P. 5196. - DOI: 10.3390/s20185196.

17. Hidalgo-Fort E. Battery-Less Industrial Wireless Monitoring and Control System for Improved Operational Efficiency / E. Hidalgo-Fort, J. A. Gómez-Galán, R. González-Carvajal та ін. // Sensors. - 2023. - Vol. 23, № 5. - P. 2517. - DOI: 10.3390/s23052517.

18. Maxim Integrated. DS18B20 Programmable Resolution 1-Wire Digital Thermometer : datasheet [Електронний ресурс]. — Maxim Integrated Products, 2019. - 20 с. - Режим доступу: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

19. NXP Semiconductors. MPX5700 Series Integrated Silicon Pressure Sensor : datasheet [Електронний ресурс]. - NXP Semiconductors, 2020. - 10 с. - Режим доступу: https://www.nxp.com/docs/en/data-sheet/MPX5700.pdf

20. Analog Devices. ADXL345 : 3-Axis, ±2 g/±4 g/±8 g/±16 g Digital Accelerometer : datasheet [Електронний ресурс]. - Analog Devices, 2015. - 36 с. - Режим доступу: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf

21. Texas Instruments. INA219 Zerø-Drift, Bidirectional Current/Power Monitor With I²C Interface : datasheet [Електронний ресурс]. - Texas Instruments Incorporated, 2015. - 35 с. - Режим доступу: https://www.ti.com/lit/ds/symlink/ina219.pdf

22. Marin-Garcia I. Sensor Node Network for Remote Moisture Measurement in Timber Based on Bluetooth Low Energy and Web-Based Monitoring System / I. Marin-Garcia, A. Costa-Castelló, H. Voos // Sensors. - 2021. - Vol. 21, № 2. - P. 491. - DOI: 10.3390/s21020491.

23. Microchip Technology. ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash : datasheet [Електронний ресурс]. - Microchip Technology Inc., 2018. - 442 с. - Режим доступу:

https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

24. Python Software Foundation. PySerial Documentation [Електронний ресурс]. - PySerial Project, 2023. - Режим доступу: https://pyserial.readthedocs.io/en/latest

25. García L. A Study on the Use of Raspberry Pis Compared to Standard PCs in a Smart Campus Application with a BLE-Based Indoor Positioning System / L. García, J. Tomás, L. Parra, J. Lloret // Sensors. - 2019. - Vol. 19, № 21. - P. 4774. - DOI: 10.3390/s19214774.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема: Розроблення системи телеметрії на базі Bluetooth-модуля CC2541

Обсяг ПЗ складає 108 аркушів

Перелік креслень графічної частини:

- КРБ.СІ-10.00.00.000Е1 – Структурна схема модуля (аркушів 1);
- КРБ.СІ-10.00.00.000Е2 – Схема функціональна (аркушів 1);
- КРБ.СІ-10.00.00.001 – Алгоритм роботи системи (аркушів 1);
- КРБ.СІ-10.00.00.002 – Фото тест-плати (аркушів 1);

Дата закінчення виконання бакалаврської роботи: _____

Студент-дипломник _____ Тимофійчук О.С.

ДОДАТОК А
Лістинг програми мікроконтролера

```
// ── Підключення бібліотек ──────────────────────────────────
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <Adafruit_ADXL345_U.h>
#include <Adafruit_INA219.h>
#include <SoftwareSerial.h>

// ── Конфігурація виводів та констант ──────────────────────
#define ONE_WIRE_BUS  2     // DS18B20 — вивід D2
#define BT_RX_PIN     2     // HM-10 TX → Arduino D2
#define BT_TX_PIN     3     // HM-10 RX ← Arduino D3
#define PRESSURE_PIN  A0    // MPX5700 — аналоговий вхід A0
#define SAMPLE_MS     1000  // інтервал опитування, мс
#define PKT_SIZE      19    // розмір пакету, байт
#define NODE_ID       0x01  // ідентифікатор вузла

// ── Об'єкти периферії ──────────────────────────────────────
OneWire           oneWire(ONE_WIRE_BUS);
DallasTemperature ds18b20(&oneWire);
Adafruit_ADXL345_Unified accel(12345);
Adafruit_INA219   ina219;
SoftwareSerial    BTSerial(BT_RX_PIN, BT_TX_PIN);

bool bleConnected = false;

void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600);

  ds18b20.begin();
  if (!accel.begin())  Serial.println("ADXL345 не знайдено!");
  if (!ina219.begin()) Serial.println("INA219 не знайдено!");

  delay(500);
  BTSerial.print("AT");              delay(100);
  BTSerial.print("AT+ROLE0");        delay(100);
  BTSerial.print("AT+NAMETelemetry"); delay(100);
  BTSerial.print("AT+NOTI1");        delay(100);

  Serial.println("Ініціалізація завершена. Очікування BLE...");
```

```
}

void loop() {
  if (BTSerial.available()) {
    String msg = BTSerial.readStringUntil('\n');
    if (msg.indexOf("OK+CONN") >= 0) {
      bleConnected = true;
      Serial.println("BLE: з'єднання встановлено");
    }
    if (msg.indexOf("OK+LOST") >= 0) {
      bleConnected = false;
      Serial.println("BLE: з'єднання розірвано");
    }
  }

  if (!bleConnected) return;

  float   temp = readTemperature();
  float   pres = readPressure();
  int16_t ax, ay, az;
  readAccel(&ax, &ay, &az);
  float volt = ina219.getBusVoltage_V();
  float curr = ina219.getCurrent_mA();

  Serial.print("T="); Serial.print(temp);
  Serial.print(" P="); Serial.print(pres);
  Serial.print(" AX="); Serial.print(ax);
  Serial.print(" V="); Serial.print(volt);
  Serial.print(" I="); Serial.println(curr);

  uint8_t pkt[PKT_SIZE];
  buildPacket(pkt, temp, pres, ax, ay, az, volt, curr);
  BTSerial.write(pkt, PKT_SIZE);

  delay(SAMPLE_MS);
}

float readTemperature() {
  ds18b20.requestTemperatures();
  float t = ds18b20.getTempCByIndex(0);
  if (t == DEVICE_DISCONNECTED_C) {
    Serial.println("Помилка: DS18B20 не відповідає");
    return -999.0;
  }
```

```
  return t;
}

float readPressure() {
  long sum = 0;
  for (int i = 0; i < 10; i++) {
    sum += analogRead(PRESSURE_PIN);
    delay(2);
  }
  float adcVal = sum / 10.0;
  float vout = (adcVal / 1023.0) * 5.0;
  float pres = (vout / 5.0 - 0.04) / 0.0012858;
  return pres;
}

void readAccel(int16_t *ax, int16_t *ay, int16_t *az) {
  sensors_event_t event;
  accel.getEvent(&event);
  *ax = (int16_t)(event.acceleration.x / 9.81 * 1000);
  *ay = (int16_t)(event.acceleration.y / 9.81 * 1000);
  *az = (int16_t)(event.acceleration.z / 9.81 * 1000);
}

void readPower(float *voltage, float *current, float *power) {
  *voltage = ina219.getBusVoltage_V();
  *current = ina219.getCurrent_mA();
  *power   = (*voltage) * (*current) / 1000.0;
  if (*current < 0) *current = 0;
}

#define NODE_ID  0x01

void buildPacket(uint8_t *pkt, float temp, float pres,
                 int16_t ax, int16_t ay, int16_t az,
                 float volt, float curr) {

  int16_t iTemp = (int16_t)(temp * 10);
  int16_t iPres = (int16_t)(pres * 10);
  int16_t iVolt = (int16_t)(volt * 1000);
  int16_t iCurr = (int16_t)(curr * 10);

  pkt[0]  = 0xAA;
  pkt[1]  = NODE_ID;
```

```
  pkt[2]  = iTemp & 0xFF;  pkt[3]  = (iTemp >> 8) & 0xFF;
  pkt[4]  = iPres & 0xFF;  pkt[5]  = (iPres >> 8) & 0xFF;
  pkt[6]  = ax    & 0xFF;  pkt[7]  = (ax    >> 8) & 0xFF;
  pkt[8]  = ay    & 0xFF;  pkt[9]  = (ay    >> 8) & 0xFF;
  pkt[10] = az    & 0xFF;  pkt[11] = (az    >> 8) & 0xFF;
  pkt[12] = iVolt & 0xFF;  pkt[13] = (iVolt >> 8) & 0xFF;
  pkt[14] = iCurr & 0xFF;  pkt[15] = (iCurr >> 8) & 0xFF;

  uint8_t crc = 0;
  for (int i = 0; i < 16; i++) crc ^= pkt[i];
  pkt[16] = crc;
  pkt[17] = 0x00;
  pkt[18] = 0x55;
}

bool parsePacket(uint8_t *pkt) {
  if (pkt[0] != 0xAA || pkt[18] != 0x55) return false;
  uint8_t crc = 0;
  for (int i = 0; i < 16; i++) crc ^= pkt[i];
  if (crc != pkt[16]) return false;
  float temp = ((int16_t)(pkt[3] << 8 | pkt[2])) / 10.0;
  float pres = ((int16_t)(pkt[5] << 8 | pkt[4])) / 10.0;
  Serial.print("T="); Serial.print(temp);
  Serial.print(" P="); Serial.println(pres);
  return true;
}
                                        }
```

ДОДАТОК Б
Лістинг програми-клієнта

```
# config.py — налаштування програми-клієнта
# Параметри послідовного порту
COM_PORT    = "COM5"
BAUD_RATE   = 9600
TIMEOUT     = 1.0

# Параметри пакету
PKT_SIZE    = 19
PKT_START   = 0xAA
PKT_END     = 0x55
NODE_ID     = 0x01

# Збереження даних
LOG_FILE    = "telemetry_log.csv"
MAX_POINTS  = 60
UPDATE_MS   = 100
```

```
import serial, threading, queue
from config import COM_PORT, BAUD_RATE, TIMEOUT, PKT_SIZE, PKT_START,
PKT_END

class SerialReader:
    def __init__(self):
        self.port      = None
        self.running   = False
        self.pkt_queue = queue.Queue()
        self._buf      = bytearray()

    def start(self):
        self.port    = serial.Serial(COM_PORT, BAUD_RATE,
timeout=TIMEOUT)
        self.running = True
        self._thread = threading.Thread(target=self._read_loop,
daemon=True)
        self._thread.start()
        print(f"Підключено до {COM_PORT} @ {BAUD_RATE} бод")

    def stop(self):
        self.running = False
        if self.port and self.port.is_open:
            self.port.close()
```

```
    def _read_loop(self):
        while self.running:
            try:
                data = self.port.read(self.port.in_waiting or 1)
                if data:
                    self._buf.extend(data)
                    self._extract_packets()
            except serial.SerialException as e:
                print(f"Помилка порту: {e}")
                self.running = False

    def _extract_packets(self):
        while len(self._buf) >= PKT_SIZE:
            idx = self._buf.find(PKT_START)
            if idx == -1:
                self._buf.clear()
                return
            if idx > 0:
                del self._buf[:idx]
            if len(self._buf) >= PKT_SIZE:
                if self._buf[PKT_SIZE - 1] == PKT_END:
                    pkt = bytes(self._buf[:PKT_SIZE])
                    self.pkt_queue.put(pkt)
                del self._buf[:PKT_SIZE]
```

```
import struct
from config import PKT_START, PKT_END

def parse_packet(pkt: bytes) -> dict | None:
    if pkt[0] != PKT_START or pkt[18] != PKT_END:
        print("Помилка: невірні маркери пакету")
        return None

    crc_calc = 0
    for i in range(16):
        crc_calc ^= pkt[i]
    if crc_calc != pkt[16]:
        print(f"Помилка CRC: {pkt[16]:02X} / {crc_calc:02X}")
        return None

    _, node_id, \
    raw_temp, raw_pres, \
    ax, ay, az, \
    raw_volt, raw_curr, \
    crc, _ = struct.unpack('<BBhhhhhhhBB', pkt[:18] + b'\x00')
```

```
    return {
        "node_id":  node_id,
        "temp":     raw_temp  / 10.0,
        "pressure": raw_pres  / 10.0,
        "accel_x":  ax,
        "accel_y":  ay,
        "accel_z":  az,
        "voltage":  raw_volt  / 1000.0,
        "current":  raw_curr  / 10.0,
        "power":    (raw_volt/1000.0)*(raw_curr/10.0)/1000.0,
    }
```

#data_logger.py

```
import csv, os
from datetime import datetime
from config import LOG_FILE

class DataLogger:
    FIELDS = ["timestamp", "node_id", "temp", "pressure",
              "accel_x", "accel_y", "accel_z",
              "voltage", "current", "power"]

    def __init__(self):
        self._file   = None
        self._writer = None
        self.count   = 0

    def open(self, filename: str = LOG_FILE):
        write_header = not os.path.exists(filename)
        self._file   = open(filename, "a", newline="", encoding="utf-8")
        self._writer = csv.DictWriter(self._file,
fieldnames=self.FIELDS)
        if write_header:
            self._writer.writeheader()

    def log(self, data: dict):
        row = {"timestamp": datetime.now().isoformat()} | data
        self._writer.writerow(row)
        self._file.flush()
        self.count += 1
```

```
    def close(self):
        if self._file:
            self._file.close()
            print(f"Лог закрито. Записано рядків: {self.count}")
```

#main.py

```
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
from collections import deque
from serial_reader import SerialReader
from packet_parser import parse_packet
from data_logger import DataLogger
from config import MAX_POINTS, UPDATE_MS

class TelemetryApp:
    def __init__(self, root):
        self.root    = root
        self.root.title("Система телеметрії BLE — HM-10 / CC2541")
        self.reader  = SerialReader()
        self.logger  = DataLogger()
        self.buf_temp = deque([0.0] * MAX_POINTS, maxlen=MAX_POINTS)
        self.buf_pres = deque([0.0] * MAX_POINTS, maxlen=MAX_POINTS)
        self.buf_curr = deque([0.0] * MAX_POINTS, maxlen=MAX_POINTS)
        self.pkt_count = 0
        self.err_count = 0
        self._build_ui()
        self._build_charts()

    def _build_ui(self):
        left = tk.Frame(self.root, width=200, bg="#F5F9FF")
        left.pack(side=tk.LEFT, fill=tk.Y, padx=5, pady=5)
        self.lbl_temp = tk.Label(left, text="— \u00b0C",
            font=("Arial", 18, "bold"), fg="#856404", bg="#F5F9FF")
        self.lbl_pres = tk.Label(left, text="— \u043a\u041f\u0430",
            font=("Arial", 18, "bold"), fg="#CC0000", bg="#F5F9FF")
        self.lbl_volt = tk.Label(left, text="— \u0412",
            font=("Arial", 18, "bold"), fg="#1E5C1E", bg="#F5F9FF")
        self.lbl_curr = tk.Label(left, text="— \u043c\u0410",
            font=("Arial", 18, "bold"), fg="#1E5C1E", bg="#F5F9FF")
        for lbl in [self.lbl_temp, self.lbl_pres,
                    self.lbl_volt, self.lbl_curr]:
```

```
            lbl.pack(pady=8, padx=10, anchor="w")
        tk.Button(left, text="\u25b6 \u0421\u0442\u0430\u0440\u0442",
            bg="#1E5C1E", fg="white",
            command=self.start).pack(fill=tk.X, padx=5, pady=2)
        tk.Button(left, text="\u25a0 \u0421\u0442\u043e\u043f",
            bg="#CC0000", fg="white",
            command=self.stop).pack(fill=tk.X, padx=5, pady=2)
        tk.Button(left, text="\U0001f4be
\u0417\u0431\u0435\u0440\u0435\u0433\u0442\u0438",
            bg="#4472C4", fg="white",
            command=self.save).pack(fill=tk.X, padx=5, pady=2)

    def _build_charts(self):
        self.fig, (self.ax1, self.ax2, self.ax3) = \
            plt.subplots(3, 1, figsize=(8, 6))
        self.fig.tight_layout(pad=2.0)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.root)
        self.canvas.get_tk_widget().pack(
            side=tk.RIGHT, fill=tk.BOTH, expand=True)

    def _update(self):
        while not self.reader.pkt_queue.empty():
            raw_pkt = self.reader.pkt_queue.get()
            data    = parse_packet(raw_pkt)
            if data:
                self.pkt_count += 1
                self.logger.log(data)
                self._refresh_labels(data)
                self.buf_temp.append(data["temp"])
                self.buf_pres.append(data["pressure"])
                self.buf_curr.append(data["current"])
            else:
                self.err_count += 1

        for ax, buf, title, color in [
            (self.ax1, self.buf_temp, "Температура (\u00b0C)",
"#D4910A"),
            (self.ax2, self.buf_pres, "Тиск (кПа)",
"#CC0000"),
            (self.ax3, self.buf_curr, "Струм (мА)",
"#1E5C1E"),
        ]:
            ax.clear()
            ax.plot(list(buf), color=color, linewidth=1.5)
            ax.set_title(title, fontsize=9)
```

```
            ax.grid(True, alpha=0.3)

        self.canvas.draw()
        self.root.title(
            f"Телеметрія BLE  |  Пакетів: {self.pkt_count}"
            f"  |  Помилок CRC: {self.err_count}"
        )
        self.root.after(UPDATE_MS, self._update)

    def _refresh_labels(self, d):
        self.lbl_temp.config(text=f"{d['temp']:.1f} \u00b0C")
        self.lbl_pres.config(text=f"{d['pressure']:.1f}
\u043a\u041f\u0430")
        self.lbl_volt.config(text=f"{d['voltage']:.2f} \u0412")
        self.lbl_curr.config(text=f"{d['current']:.1f} \u043c\u0410")

    def start(self):
        self.reader.start()
        self.logger.open()
        self._update()

    def stop(self):
        self.reader.stop()
        self.logger.close()

    def save(self):
        print(f"Збережено {self.logger.count} записів")

if __name__ == "__main__":
    root = tk.Tk()
    app  = TelemetryApp(root)
    root.mainloop()
```

```
# Встановлення необхідних бібліотек
pip install pyserial matplotlib

# Запуск програми-клієнта
python main.py
```