

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

БАРАНОВ Сергій Олексійович

**Програмно-апаратний модуль реального часу для збору даних сенсорів на
ESP32 / Real-Time Hardware-Software Module for Sensor Data Acquisition on
ESP32**

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
С. О. Баранов

Науковий керівник:
к.т.н., доцент, О.Р. Осолінський

Кваліфікаційну роботу допущено до
захисту

«___»_____ 2026 р.

В.о. завідувача кафедри
_____ Н.В. Дзюбановська

Тернопіль – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Н.В. Дзюбановська

«_____» _____ 20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
БАРАНОВ Сергій Олексійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: **Програмно-апаратний модуль реального часу для збору даних сенсорів на ESP32 / Real-Time Hardware-Software Module for Sensor Data Acquisition on ESP32**

керівник роботи О.Р. Осолінський, к.т.н., доцент

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

затверджені наказом по університету від 01 грудня 2025 р. № 894.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
 - провести аналіз предметної області систем збору даних реального часу та огляд існуючих рішень на базі мікроконтролера ESP32;
 - дослідити проблему одночасного опитування різнотипних сенсорів (I2C, 1-Wire, ADC) при активному Bluetooth-стеку та визначити вимоги до архітектури програмного забезпечення;
 - спроектувати трирівневу архітектуру програмного забезпечення на основі FreeRTOS з розподілом задач між двома ядрами ESP32;
 - реалізувати драйвери сенсорів DHT22, BMP280, MPU6050 та MQ-135 з уніфікованим інтерфейсом та алгоритмами калібрування;
 - розробити компактний бінарний протокол передачі даних через Bluetooth SPP з контролем цілісності CRC-8/MAXIM;
 - провести дослідження часових характеристик модуля та верифікацію точності вимірювань усіх чотирьох сенсорів.
5. Перелік графічного матеріалу в роботі:
 - загальна архітектура програмно-апаратного модуля;
 - алгоритми опитування та калібрування сенсорів;

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01.2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02.2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент _____ С.О.Баранов
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи _____ О.Р. Осолінський
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмно-апаратний модуль реального часу для збору даних сенсорів на ESP32» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньо-професійної програми «Комп'ютерні науки» написана обсягом в 53 сторінок і містить 11 ілюстрацій, 6 таблиці, 5 додатків та 29 використаних джерел.

Метою роботи є розроблення та дослідження програмно-апаратного модуля реального часу для збору даних з різнотипних сенсорів на базі мікроконтролера ESP32 з бездротовою передачею даних через інтерфейс Bluetooth.

Методами дослідження обрано: аналіз науково-технічної літератури та документації; структурне та об'єктно-орієнтоване проєктування програмного забезпечення; порівняльний аналіз для оцінювання існуючих рішень; експериментальний метод для перевірки характеристик розробленого модуля; методи тестування вбудованих програмних систем.

В результаті виконання роботи розроблено трирівневу архітектуру програмного забезпечення на основі FreeRTOS з розподілом задач між двома ядрами ESP32, реалізовано уніфікований інтерфейс драйверів сенсорів DHT22, BMP280, MPU6050 та MQ-135, розроблено компактний бінарний протокол передачі даних через Bluetooth з контролем цілісності CRC-8. Підтверджено, що джитер опитування MPU6050 у режимі активного Bluetooth становить $\pm 0,74$ мс, що у 8–40 разів краще порівняно з аналогічними рішеннями.

Результати роботи можуть бути використані у системах моніторингу мікроклімату, промислової автоматизації, носимої електроніки та «розумного будинку».

Ключові слова: ESP32, FREERTOS, ЗБІР ДАНИХ СЕНСОРІВ, BLUETOOTH, СИСТЕМИ РЕАЛЬНОГО ЧАСУ, МІКРОКОНТРОЛЕР, ВБУДОВАНІ СИСТЕМИ, IoT.

ANNOTATION

Qualification work on the topic «Real-Time Hardware-Software Module for Sensor Data Acquisition on ESP32» for the Bachelor's degree in speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 53 pages and contains 11 figures, 6 tables, 5 annexes and 29 sources.

The purpose of the work is the development and study of a real-time hardware-software module for collecting data from various types of sensors based on the ESP32 microcontroller with wireless data transmission via the Bluetooth interface.

The research methods are: analysis of scientific and technical literature and documentation; structural and object-oriented software design; comparative analysis for evaluating existing solutions; the experimental method for verifying the characteristics of the developed module; testing methods for embedded software systems.

As a result of the work, a three-tier software architecture based on FreeRTOS was developed with task distribution between the two cores of the ESP32, a unified driver interface for the DHT22, BMP280, MPU6050 and MQ-135 sensors was implemented, and a compact binary data transmission protocol over Bluetooth with CRC-8 integrity control was developed. It was confirmed that the MPU6050 polling jitter in active Bluetooth mode is $\pm 0,74$ ms, which is 8–40 times better compared to similar solutions.

The results of the work can be used in microclimate monitoring systems, industrial automation, wearable electronics and «smart home» systems.

Keywords: ESP32, FREERTOS, SENSOR DATA ACQUISITION, BLUETOOTH, REAL-TIME SYSTEMS, MICROCONTROLLER, EMBEDDED SYSTEMS, IoT.

ЗМІСТ

Вступ.....	7
1 Аналіз існуючих рішень та обґрунтування теми дослідження.....	9
1.1 Аналіз предметної області систем збору даних реального часу	9
1.2 Огляд існуючих рішень та протоколів передачі даних для IoT-систем	12
1.3 Опис недоліків існуючих рішень та формування завдань дослідження	14
2 Проектування архітектури та алгоритмічне забезпечення модуля.....	16
2.1 Архітектура модуля та схема підключення сенсорів	16
2.2 Алгоритми опитування та калібрування сенсорів	19
2.3 Протокол передачі даних та методика дослідження характеристик модуля .	21
3 Програмна реалізація та результати досліджень	24
3.1 Реалізація програмного забезпечення та ініціалізація системи	24
3.2 Реалізація задач FreeRTOS та підсистеми Bluetooth	26
3.3 Інтерфейс користувача та тестування програмного забезпечення	28
3.4 Результати дослідження часових характеристик та точності вимірювань	30
Висновки	34
Список використаних джерел	37
Додаток А Архітектура програмного модуля	40
Додаток Б Основні алгоритми програмного модуля	41
Додаток В Програмна реалізація основних алгоритмів	42
Додаток Г Дані, які використовувались у програмі	47
Додаток Д Апробація отриманих результатів	48

ВСТУП

Розвиток Інтернету речей та вбудованих систем зумовлює потребу в надійних, енергоефективних та масштабованих рішеннях для збору і передачі даних від різнотипних сенсорів в режимі реального часу. Системи моніторингу довкілля, промислової автоматизації та носимої електроніки вимагають апаратно-програмних платформ, здатних забезпечити стабільне опитування сенсорів, мінімальну затримку обробки та бездротову передачу даних.

Мікроконтролер ESP32 є одним із найпоширеніших рішень для побудови IoT-пристроїв завдяки інтегрованим модулям, широкому набору периферійних інтерфейсів та відносно низькій вартості. Проте реалізація системи збору даних з кількох різнотипних сенсорів одночасно потребує проєктування архітектури програмного забезпечення, управління задачами на основі FreeRTOS та оптимізації протоколів зв'язку.

Актуальність роботи полягає в побудові багатосенсорних систем збору даних реального часу на базі ESP32 з передачею через Bluetooth, а також необхідністю дослідження часових характеристик, точності та надійності таких систем при одночасній роботі з сенсорами температури й вологості - DHT22, атмосферного тиску - BMP280, інерційними вимірювальними сенсорами - MPU6050 та якості повітря - MQ-135.

Мета роботи — розробити та дослідити програмно-апаратний модуль реального часу для збору даних з різнотипних сенсорів на базі мікроконтролера ESP32 з бездротовою передачею даних через інтерфейс Bluetooth.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати існуючі підходи до побудови систем збору даних на базі мікроконтролерів ESP32 та виявити їхні переваги і недоліки;
- дослідити характеристики обраних сенсорів DHT22, BMP280, MPU6050, MQ-135 та їхні інтерфейси підключення до ESP32;
- розробити архітектуру програмного забезпечення модуля на основі FreeRTOS із забезпеченням роботи в режимі реального часу;

- реалізувати програмний модуль збору, обробки та передачі даних через Bluetooth на мові C++ в середовищі Arduino IDE / ESP-IDF; провести експериментальне дослідження часових характеристик, точності вимірювань та стабільності роботи розробленого модуля.

Об'єктом дослідження є процеси збору, обробки та передачі даних від різнотипних сенсорів у вбудованих системах реального часу.

Предметом дослідження є методи та засоби побудови програмно-апаратного модуля реального часу для збору даних сенсорів на базі мікроконтролера ESP32 з передачею через Bluetooth Low Energy.

Методи дослідження: аналіз науково-технічної літератури та документації; структурне та об'єктно-орієнтоване проєктування програмного забезпечення; метод порівняльного аналізу для оцінювання існуючих рішень; експериментальний метод для перевірки характеристик розробленого модуля; методи тестування вбудованих програмних систем.

Практичне значення одержаних результатів. Розроблений програмно-апаратний модуль може бути використаний у системах моніторингу мікроклімату, промислової автоматизації, носимої електроніки. Результати дослідження часових характеристик є корисними для розробників IoT-систем на базі ESP32.

Апробація результатів. Основні положення та результати роботи апробовано на X Міжнародній студентській науковій конференції “Тренди та перспективи розвитку мультидисциплінарних досліджень” (05.06.2026, м. Івано-Франківськ, Україна).

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області систем збору даних реального часу

Системи збору даних реального часу є ключовим компонентом сучасних IoT-рішень, промислової автоматизації, медичної діагностики та моніторингу навколишнього середовища. Під режимом реального часу розуміють режим функціонування системи, за якого результати обробки даних формуються протягом визначеного часового інтервалу, достатнього для впливу на процес, що контролюється. Відповідно до класифікації систем реального часу, розрізняють жорсткі — hard real-time та м'які — soft real-time системи. В жорстких системах порушення часових обмежень є неприпустимим і призводить до критичних відмов. М'які системи допускають певні відхилення від часових меж без катастрофічних наслідків, але з деградацією якості обслуговування. Система збору даних сенсорів на базі ESP32 відноситься до класу м'яких систем реального часу, тому що незначні відхилення в часі опитування сенсорів не призводять до критичних наслідків, але впливають на точність часових рядів вимірювань. Основними характеристиками систем RT-DAQ є:

- частота дискретизації, тобто скільки вимірювань за одиницю часу здатна виконати система;
- часова затримка між фізичною подією та її реєстрацією системою;
- варіація затримки між послідовними вимірами;
- пропускна здатність каналу передачі даних.

Ці параметри є взаємопов'язаними і визначають придатність системи для конкретного застосування. Сучасний ринок вбудованих систем пропонує широкий спектр платформ для реалізації RT-DAQ: від 8-бітних мікроконтролерів сімейства AVR до 32-бітних ARM-систем та SoC із підтримкою RTOS. Вибір платформи визначається вимогами до обчислювальної потужності, енергоспоживання, набором периферійних інтерфейсів та засобами бездротового зв'язку. Мікроконтролер ESP32, завдяки поєднанню двоядерного процесора, інтегрованого Bluetooth/Wi-Fi та підтримки FreeRTOS, займає оптимальну нішу для побудови IoT-систем збору даних середньої складності.

ESP32 — це мікроконтролер компанії Espressif Systems, представлений у 2016 році як наступник ESP8266. SoC побудований на двоядерному процесорі, що забезпечує достатню обчислювальну потужність для паралельної обробки даних від кількох сенсорів. Архітектурні особливості ESP32 наведено в таблиці 1.1.

Таблиця 1.1 – Технічні характеристики мікроконтролера ESP32

Характеристика	Значення
Процесор	Xtensa LX6 dual-core, до 240 МГц
RAM	520 КБ SRAM
Flash	4–16 МБ (зовнішня)
Wi-Fi	802.11 b/g/n, 2.4 ГГц
Bluetooth	BT Classic та BLE 4.2
GPIO	34 програмовані виводи
ADC	12-бітний, 18 каналів
Інтерфейси	I2C, SPI, UART, I2S, CAN, PWM
RTOS	FreeRTOS (вбудований)
Напруга живлення	2,2–3,6 В

Перевагою ESP32 для задач RT-DAQ є вбудована підтримка операційної системи реального часу FreeRTOS. FreeRTOS забезпечує детермінований планувальник задач з пріоритетами, механізми міжзадачної комунікації та таймери програмного рівня. Це дозволяє організувати паралельне опитування сенсорів на різних ядрах процесора з чітко визначеними часовими інтервалами. Двоядерна архітектура ESP32 дає можливість розподілу задач між ядрами. Перше ядро під назвою Core0 може бути відповідальним за мережеву підсистему - друге Core1 за опитування сенсорів та обробку даних. Такий розподіл мінімізує вплив комунікаційного стеку на часові характеристики збору даних, що є критичним для систем реального часу. Серед обмежень платформи є нелінійність АЦП при напрузі вище 3,1 В та нижче 0,1 В, що потребує програмної калібрації при використанні аналогових сенсорів. Крім того, одночасне використання Wi-Fi та ADC може призводити до перешкод через спільне живлення, що вирішується використанням зовнішнього джерела опорної напруги або перемиканням режимів роботи. Для

реалізації багатопараметричного модуля збору даних було обрано чотири типи сенсорів, що охоплюють різні фізичні явища та використовують різні інтерфейси підключення до ESP32. Такий вибір дозволяє дослідити особливості одночасної роботи з сенсорами різних типів в єдиній системі реального часу. Сенсор DHT22 є цифровим сенсором температури та відносної вологості повітря виробництва компанії AOSONG. Діапазон вимірювання температури становить від -40°C до $+80^{\circ}\text{C}$ з точністю $\pm 0,5^{\circ}\text{C}$, вологості — від 0% до 100% відносної вологості з точністю $\pm 2-5\%$. Сенсор використовує однопровідний послідовний інтерфейс для передачі 40-бітних пакетів даних, 16 біт вологості + 16 біт температури + 8 біт контрольної суми. Мінімальний інтервал опитування становить 2 секунди, що є обмеженням для систем з високою частотою дискретизації.

Сенсор BMP280 від компанії Bosch Sensortec є цифровим барометричним сенсором тиску та температури. Діапазон вимірювання тиску: 300–1100 гПа з відносною точністю $\pm 0,12$ гПа, температури: -40°C до $+85^{\circ}\text{C}$ з точністю $\pm 0,5^{\circ}\text{C}$. Сенсор підтримує інтерфейси I2C та SPI, що забезпечує гнучкість підключення. BMP280 має вбудовані фільтри IIR- Infinite Impulse Response, для згладжування шумів вимірювань. Частота виведення даних може досягати 157 Гц, що перевищує потреби більшості IoT-застосувань.

Сенсор MPU6050. MPU6050 виробництва компанії InvenSense є шестиосьовим інерційним вимірювальним пристроєм, що поєднує тривісний акселерометр та тривісний гіроскоп. Акселерометр вимірює прискорення в діапазоні $\pm 2\text{g}/\pm 4\text{g}/\pm 8\text{g}/\pm 16\text{g}$, гіроскоп — кутову швидкість в діапазоні $\pm 250/\pm 500/\pm 1000/\pm 2000$ $^{\circ}/\text{с}$. Обидва сенсори мають 16-бітні АЦП. MPU6050 використовує інтерфейс I2C та має вбудований цифровий процесор руху DMP — Digital Motion Processor, який може виконувати попередню обробку даних. Частота виведення даних становить 1 кГц для гіроскопа та 1 кГц для акселерометра.

Сенсор MQ-135. MQ-135 є аналоговим резистивним сенсором газу, чутливим до аміаку, бензолу, спирту, диму та діоксиду вуглецю. Робочий діапазон: 10–1000 ppm для NH_3 , 10–300 ppm для бензолу. Сенсор видає аналоговий сигнал напруги від 0 В до 5 В, що потребує підключення до АЦП ESP32. Потужність нагрівального елемента становить близько 0.9 Вт, що потребує часу прогріву приблизно 20–30

секунд для стабільних показників. Для перетворення аналогового сигналу в значення концентрації газу використовується апроксимація логарифмічної характеристики сенсора.

Порівняльна характеристика обраних сенсорів за ключовими параметрами наведена в таблиці 1.2.

Таблиця 1.2 – Порівняння характеристик обраних сенсорів

Сенсор	Параметр	Інтерфейс	Частота	Розрядність
DHT22	Темп., вологість	1-Wire	0,5 Гц	16 біт
BMP280	Тиск, температура	I2C/SPI	157 Гц	20 біт
MPU6050	Акселер., гіроскоп	I2C	1000 Гц	16 біт
MQ-135	Якість повітря	ADC	до 100 Гц	12 біт

1.2 Огляд існуючих рішень та протоколів передачі даних для IoT-систем

Бібліотека ESPHome — це фреймворк конфігурування прошивки ESP32/ESP8266 на основі файлів YAML. Підтримує широкий спектр сенсорів, включаючи всі чотири обрані в даній роботі. ESPHome автоматично генерує C++ код та забезпечує інтеграцію з системами автоматизації Home Assistant. Перевагою є висока швидкість розгортання та простота налаштування. Недоліком є обмежена гнучкість у налаштуванні часових параметрів опитування сенсорів та відсутність можливості тонкого управління пріоритетами задач FreeRTOS.

Платформа Arduino framework for ESP32 ESP32 Arduino Core забезпечує сумісний з Arduino API рівень абстракції для ESP32. Це найпоширеніший підхід для прототипування IoT-пристроїв. Перевагами є велика спільнота, велика кількість бібліотек та доступна документація. Критичним недоліком є те, що Arduino loop() виконується як задача FreeRTOS з фіксованим пріоритетом, що ускладнює реалізацію багатозадачності з різними часовими характеристиками.

ESP-IDF - це офіційний фреймворк від Espressif Systems надає повний доступ до можливостей ESP32, включаючи безпосереднє управління задачами FreeRTOS, прив'язку задач до конкретного ядра та використання апаратних таймерів. ESP-IDF забезпечує максимальну гнучкість та детермінованість виконання. Недоліком є

значно вища складність розробки порівняно з Arduino API та менший вибір готових бібліотек для конкретних сенсорів.

MicroPython для ESP32 — це інтерпретована мова програмування, оптимізована для мікроконтролерів. Перевагою є швидкість розробки та читабельність коду. Але інтерпретатор вносить значний джитер у часові характеристики порядку десятків мілісекунд, що є неприйнятним для систем збору даних з жорсткими вимогами до синхронізації.

В таблиці 1.3 представлено порівняння засобів розробки

Таблиця 1.3 – Порівняльний аналіз підходів до розробки програмного забезпечення для ESP32

Підхід	Гнучкість	RT-контроль	Складність	Бібліотеки
ESPHome	Низька	Низький	Низька	Обмежені
Arduino/ESP32	Середня	Середній	Середня	Широкі
ESP-IDF	Висока	Високий	Висока	Обмежені
MicroPython	Середня	Низький	Низька	Середні

Вибір протоколу бездротової передачі даних є теж важливим для систем RT-DAQ, тому що він визначає затримку доставки даних, енергоспоживання, дальність зв'язку та складність реалізації. ESP32 підтримує Wi-Fi 802.11 b/g/n та Bluetooth 4.2 — Classic та BLE, що надає вибір між двома основними протоколами.

Протокол MQTT - Message Queuing Telemetry Transport, є стандартом для IoT-комунікацій поверх TCP/IP. Забезпечує надійну доставку повідомлень з рівнями якості обслуговування QoS 0, 1, 2, публікацію/підписку на топіки та підтримку посередника. Недоліком для систем реального часу є значний overhead TCP/IP стеку, необхідність підключення до мережевої інфраструктури та більше енергоспоживання — близько 160 мА при передачі.

Serial Port Profile забезпечує емуляцію послідовного порту через Bluetooth, що спрощує інтеграцію з комп'ютером або смартфоном. Підходить для передачі

потоків даних сенсорів в реальному часі без вимог до мінімального енергоспоживання.

Bluetooth Low Energy забезпечує мале енергоспоживання — близько 15 мА при передачі порівняно з Bluetooth Classic, що підходить для автономних пристроїв. BLE використовує модель GATT з сервісами та характеристиками. Максимальна пропускна здатність BLE 4.2 становить близько 250 кбіт/с корисного навантаження, що достатньо для передачі показань сенсорів з частотою до кількох десятків Гц. Затримка повідомлень залежить від connection interval та становить від 7,5 мс до кількох секунд.

Для реалізації обрано Bluetooth - Classic SPP та BLE, як основний протокол передачі даних, що обґрунтовується такими факторами: відсутність необхідності в мережевій інфраструктурі, а саме точки доступу Wi-Fi, для локального застосування; можливість прямого підключення до смартфона або ноутбука; менше втручання в часові характеристики системи реального часу порівняно з Wi-Fi стеком; достатня пропускна здатність для передачі даних усіх чотирьох сенсорів.

1.3 Опис недоліків існуючих рішень та формування завдань дослідження

На основі проведеного аналізу існуючих підходів до побудови систем збору даних сенсорів на ESP32 виявлено такі основні недоліки:

- відсутність комплексних рішень для одночасного опитування різнотипних сенсорів, а саме цифрових однопровідних, I2C та аналогових в межах єдиної багатозадачної архітектури з керованими часовими параметрами;
- недостатня дослідженість впливу конфігурації планувальника FreeRTOS при виставленні пріоритетів задач, прив'язки до ядра, визначення часових квантів на джитер та латентність при одночасній роботі кількох сенсорів;
- обмеженість існуючих бібліотек для роботи з сенсором MQ-135 в частині програмної калібрації та перетворення аналогового сигналу АЦП у значення концентрації газу;
- відсутність оцінки впливу роботи Bluetooth-стека на часові характеристики опитування сенсорів при використанні обох ядер ESP32;

- брак уніфікованих форматів пакетів даних для передачі різнотипних вимірювань через Bluetooth з мінімальним накладним трафіком.

З врахуванням виявлених недоліків та мети роботи сформульовано такі конкретні завдання дослідження:

1) розробити багатозадачну архітектуру програмного забезпечення на основі FreeRTOS з оптимальним розподілом задач між двома ядрами ESP32 для мінімізації джитеру опитування сенсорів;

2) реалізувати уніфікований інтерфейс драйверів сенсорів DHT22, BMP280, MPU6050 та MQ-135 з підтримкою асинхронного опитування через черги FreeRTOS;

3) розробити компактний бінарний протокол пакетування даних для ефективної передачі показань усіх сенсорів через Bluetooth;

4) провести вимірювання часових характеристик системи (джитер, латентність, частота опитування) та дослідити вплив Bluetooth-стека на ці характеристики;

5) верифікувати точність вимірювань обраних сенсорів та провести порівняння з еталонними значеннями.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

2.1 Архітектура модуля та схема підключення сенсорів

Програмно-апаратний модуль реального часу для збору даних сенсорів складається з двох основних частин: апаратної та програмної. Апаратна частина включає мікроконтролер ESP32 як центральний обчислювальний елемент, чотири сенсори, підключені через відповідні інтерфейси, та джерело живлення 3,3/5 В. Програмна частина реалізована на основі Arduino Core для ESP32 з прямим використанням API FreeRTOS для управління задачами реального часу. Архітектура програмного забезпечення модуля побудована за принципом багатoshарової декомпозиції та включає три рівні (додаток А):

- рівень драйверів сенсорів - забезпечує низькорівневу взаємодію з апаратними інтерфейсами та надає уніфікований API для зчитування даних;
- рівень задач реального часу - реалізує планування опитування сенсорів, буферизацію даних через черги FreeRTOS та управління пріоритетами;
- рівень комунікації - відповідає за пакетування, серіалізацію та передачу даних через Bluetooth-інтерфейс.

Розподіл задач між ядрами ESP32 є ключовим архітектурним рішенням, що забезпечує мінімальний вплив комунікаційного стека на часові характеристики збору даних. Core 0 обслуговує Bluetooth-стек та задачу передачі даних. Core 1 виконує задачі опитування сенсорів з жорсткими часовими обмеженнями. Така конфігурація усуває конкуренцію між Bluetooth-стеком та задачами сенсорів за процесорний час на одному ядрі.

На рисунку 2.1 наведено загальну структурну схему програмно-апаратного модуля. Центральне місце займає ESP32, до якого підключено сенсори через шини I2C - BMP280, MPU6050, однопровідний інтерфейс DHT22 та АЦП MQ-135. Вихідний канал — Bluetooth SPP/BLE для передачі даних на приймальний пристрій такий як смартфон чи ноутбук.

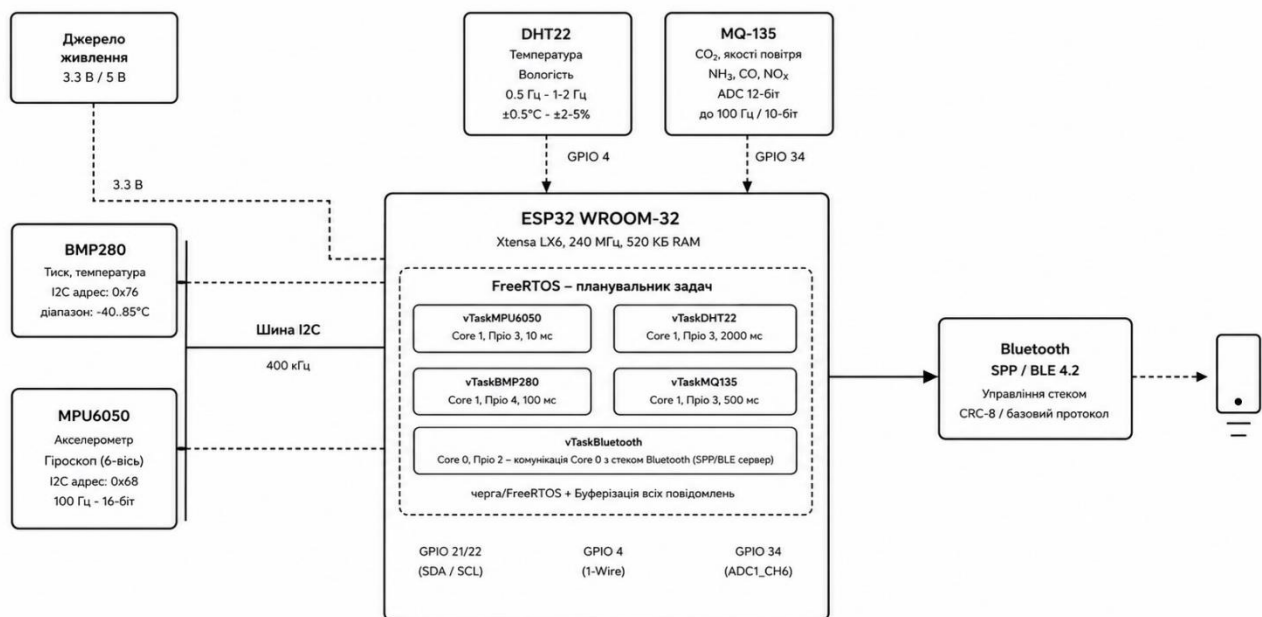


Рисунок 2.1 - Структурна схема програмно-апаратного модуля

Підключення сенсорів до ESP32 здійснюється з врахуванням електричних характеристик кожного інтерфейсу та необхідності мінімізації взаємних завад. Нижче наведено опис схем підключення кожного сенсора. Підключення BMP280 та MPU6050 шина I2C. Обидва сенсори підключаються до спільної шини I2C ESP32. Стандартні виводи I2C для ESP32: GPIO21 та GPIO22. Підтягуючі резистори 4,7 кОм підключаються від ліній SDA та SCL до шини живлення 3,3 В. Адреса BMP280 — 0x76 SDO підключено до GND, адреса MPU6050 — 0x68 AD0 підключено до GND. Швидкість шини I2C встановлюється 400 кГц Fast Mode для забезпечення достатньої пропускної здатності при опитуванні обох пристроїв.

Підключення DHT22 має однопровідний інтерфейс. Сигнальний вивід DHT22 підключається до GPIO4 ESP32 через підтягуючий резистор 10 кОм до шини 3,3 В. Живлення сенсора — 3,3 В або 5 В при живленні 5 В рівні сигналу є сумісними з 3,3 В логікою ESP32 за рахунок підтягуючого резистора. Протокол DHT22 є тайм-чутливим — маніпуляції з шиною вимагають точного вимірювання тривалості імпульсів порядку 26–28 мкс та 70 мкс.

Підключення MQ-135 через аналоговий вивід AOUT сенсора MQ-135 підключається до GPIO34 ESP32, який є входом АЦП1_CH6. Живлення

нагрівального елемента сенсора — 5 В забезпечується від роз'єму USB або зовнішнього джерела. Так як ESP32 АЦП працює з напругою 0–3,3 В, а сенсор MQ-135 може видавати до 5 В, використовується резистивний дільник напруги $R1 = 10$ кОм, $R2 = 20$ кОм для масштабування діапазону 0–5 В до 0–3,3 В.

Призначення виводів ESP32 для підключення сенсорів наведено в таблиці 2.1.

Таблиця 2.1 – Призначення виводів ESP32

GPIO	Функція	Сенсор	Примітка
GPIO4	DHT22_DATA	DHT22	Підтяжка 10 кОм до 3,3В
GPIO21	I2C_SDA	BMP280, MPU6050	Підтяжка 4,7 кОм до 3,3В
GPIO22	I2C_SCL	BMP280, MPU6050	Підтяжка 4,7 кОм до 3,3В
GPIO34	ADC1_CH6	MQ-135	Дільник 10 кОм/20 кОм
GPIO2	Статусний LED	—	Індикація роботи

FreeRTOS є операційною системою реального часу з витісняючим плануванням на основі пріоритетів. Кожна задача FreeRTOS є нескінченним циклом з визначеним стеком та пріоритетом. Планувальник активує задачу з найвищим пріоритетом, яка перебуває у стані готовності. Задачі з однаковим пріоритетом виконуються за алгоритмом Round-Robin з квантом часу 1 мс налаштовується через FreeRTOSConfig.h.

Для даного модуля визначено п'ять задач FreeRTOS, наведених у таблиці 2.2, з відповідними пріоритетами та прив'язкою до ядер процесора.

Таблиця 2.2 – Задачі FreeRTOS модуля збору даних

Задача	Пріоритет	Ядро	Період	Стек
vTaskMPU6050	5	1	10 мс	4096 байт
vTaskBMP280	4	1	100 мс	2048 байт
vTaskDHT22	3	1	2000 мс	2048 байт
vTaskMQ135	3	1	500 мс	2048 байт
vTaskBluetooth	2	0	—	8192 байт

Задача vTaskMPU6050 має найвищий пріоритет, оскільки інерційні вимірювання потребують найвищої частоти дискретизації 100 Гц для коректного відстеження динамічних процесів. Задача vTaskBMP280 опитується з частотою 10 Гц, що перекриває вимоги більшості застосувань для вимірювання тиску. Задачі vTaskDHT22 та vTaskMQ135 мають однаковий пріоритет та нижчу частоту через апаратні обмеження сенсорів. Міжзадачна комунікація реалізована через черги FreeRTOS xQueue. Кожна задача-виробник сенсорних даних записує структуру виміряних значень у відповідну чергу. Задача vTaskBluetooth є споживачем - вона зчитує дані з усіх черг та формує пакет для передачі. Довжина черг встановлена рівною 5 елементів, що забезпечує буферизацію при тимчасових затримках Bluetooth-передачі. Синхронізація доступу до спільних ресурсів реалізована через м'ютекс xSemaphoreCreateMutex. Перед зверненням до шини I2C задача захоплює м'ютекс, виконує транзакцію та звільняє м'ютекс. Максимальний час очікування м'ютексу встановлено рівним подвоєному часу I2C-транзакції приблизно 2 мс при 400 кГц та 40-байтовому пакеті.

2.2 Алгоритми опитування та калібрування сенсорів

Узагальнений алгоритм роботи задачі опитування сенсора має типову структуру, спільну для всіх чотирьох задач-виробників, з відмінностями у специфіці звернення до конкретного сенсора. Алгоритм складається з фази ініціалізації та основного циклу опитування. Фаза ініціалізації виконується одноразово при старті задачі та включає такі кроки:

1. ініціалізація апаратного інтерфейсу I2C, GPIO, ADC та бібліотеки сенсора;
2. перевірка наявності та коректності відповіді сенсора, перевірка ідентифікатора пристрою за I2C-адресою;
3. налаштування параметрів сенсора, режим роботи, діапазон вимірювання, частота виведення даних, фільтри;
4. прогрів сенсора, актуально для MQ-135 — затримка 30 секунд;
5. отримання дескриптора черги даних з реєстру черг модуля.

Основний цикл опитування є нескінченним та виконується з фіксованим періодом, що забезпечується функцією `vTaskDelayUntil()`. Використання `vTaskDelayUntil()` замість `vTaskDelay()` є принциповим — перша функція забезпечує абсолютну прив'язку до часових міток, компенсуючи час виконання задачі, тоді як друга вносить накопичувальний дрейф у часовий ряд вимірювань.

Алгоритм основного циклу для задачі опитування сенсора:

1. фіксація поточної мітки часу `xLastWakeTime = xTaskGetTickCount()`;
2. захоплення м'ютексу шини I2C, для I2C-сенсорів з таймаутом `pdMS_TO_TICKS(10)`;
3. зчитування сирих даних з регістрів сенсора;
4. звільнення м'ютексу шини I2C;
5. перетворення сирих даних у фізичні одиниці згідно з характеристичною функцією сенсора;
6. валідація отриманих значень, перевірка виходу за межі фізично допустимого діапазону;
7. формування структури `SensorData` з полями: значення, мітка часу, ідентифікатор сенсора, прапор валідності;
8. запис структури у чергу `xQueueSend()` з таймаутом 0, без очікування - якщо черга повна, пакет відкидається та інкрементується лічильник помилок;

Алгоритм опитування сенсора DHT22 має відмінність від I2C-сенсорів через особливості однопровідного протоколу. Протокол DHT22 вимагає затримки в часі для маніпуляцій з GPIO: мікроконтролер ініціює передачу сигналом LOW тривалістю не менше 1 мс, після чого сенсор відповідає послідовністю з 40 біт. Декодування бітів здійснюється шляхом вимірювання тривалості HIGH-імпульсів: менше 30 мкс відповідає логічному 0, більше 70 мкс — логічній 1. Для точного вимірювання тривалостей використовується функція `esp_timer_get_time()`, що повертає мікросекундну мітку часу на основі апаратного таймера ESP32.

Сенсор MQ-135 є резистивним газовим сенсором, опір якого R_s змінюється залежно від концентрації цільового газу. Для перетворення показань АЦП у значення концентрації необхідно виконати наступні кроки: обчислити поточний

опір сенсора R_s та застосувати характеристичну функцію для отримання концентрації.

Обчислення опору сенсора R_s виконується за формулою:

$$R_s = R_L \times (V_{cc} - V_{adc}) / V_{adc}, \quad (2.1)$$

де R_L – опір навантажувального резистора на платі модуля MQ-135, зазвичай 10 кОм;

V_{cc} – напруга живлення сенсора;

V_{adc} – напруга, виміряна АЦП ESP32 після дільника напруги з врахуванням коефіцієнта ділення.

Процедура калібрування R_0 виконується при першому запуску модуля у чистому середовищі: протягом 30 секунд прогріву усереднюється 100 послідовних вимірювань АЦП, обчислюється R_s у чистому повітрі, після чого $R_0 = R_s$.

Компенсація температури та вологості для MQ-135 виконується з використанням коефіцієнта корекції CORA, обчисленого на основі показань DHT22 за емпіричною формулою:

$$CORA = a \times T^2 + b \times T + c + (H - 33) \times (d \times T^2 + e \times T + f), \quad (2.2)$$

де T – температура, °C;

H – відносна вологість, %;

a – f – емпіричні коефіцієнти ($a = 0,00035$; $b = -0,02718$; $c = 1,39538$; $d = -0,0000369$; $e = 0,000484$; $f = 0,0042$).

Скоригований опір $R_{s_corrected} = R_s / CORA$ використовується замість R_s у подальших обчисленнях концентрації газу.

2.3 Протокол передачі даних та методика дослідження характеристик модуля

Для передачі даних від усіх чотирьох сенсорів через Bluetooth розроблено компактний бінарний протокол пакетування. Протокол забезпечує мінімальний

розмір пакету при збереженні можливості ідентифікації джерела даних, мітки часу та контролю цілісності.

Структура пакету даних наведена в таблиці 2.3.

Таблиця 2.3 – Структура пакету бінарного протоколу

Поле	Розмір	Тип	Опис
START	1 байт	uint8_t	Стартовий байт 0xAA
SENSOR_ID	1 байт	uint8_t	Ідентифікатор сенсора (0x01–0x04)
TIMESTAMP	4 байти	uint32_t	Мілісекунди від старту (millis())
PAYLOAD_LEN	1 байт	uint8_t	Довжина поля даних
PAYLOAD	N байт	—	Дані сенсора (float-значення)
CRC8	1 байт	uint8_t	Контрольна сума CRC-8/MAXIM
END	1 байт	uint8_t	Стоповий байт 0x55

Ідентифікатори сенсорів: 0x01 - DHT22 температура + вологість, $2 \times \text{float} = 8$ байт, 0x02 — BMP280 тиск + температура, $2 \times \text{float} = 8$ байт, 0x03 - MPU6050 акселерометр 3 осі + гіроскоп 3 осі, $6 \times \text{float} = 24$ байти, 0x04 - MQ-135 концентрація CO₂ в ppm, $1 \times \text{float} = 4$ байти. Мінімальний розмір пакету MQ-135 - 11 байт, максимальний MPU6050 - 33 байти.

CRC-8 обчислюється над полями SENSOR_ID, TIMESTAMP, PAYLOAD_LEN та PAYLOAD з використанням поліному 0x31, алгоритм CRC-8/MAXIM. Приймач використовує CRC для відфільтровування пошкоджених пакетів без необхідності повторної передачі, що спрощує реалізацію приймача та зменшує затримки.

Задача vTaskBluetooth реалізує алгоритм формування та відправлення пакетів:

- 1) опитування черг усіх чотирьох сенсорів з таймаутом `xQueueReceive() = 0` (неблокуюче читання);
- 2) для кожного отриманого пакету даних — серіалізація у бінарний буфер згідно зі структурою таблиці 2.3;
- 3) обчислення CRC-8 для сформованого буфера;
- 4) передача пакету через `SerialBT.write()` (Bluetooth SPP) або через BLE notify;

5) оновлення статистики: лічильники переданих пакетів, помилок CRC на стороні відправника, пропущених пакетів (queue overflow).

Для оцінювання характеристик розробленого модуля реального часу визначено методику вимірювання трьох ключових параметрів: джитеру опитування сенсорів, латентності передачі даних через Bluetooth та точності вимірювань.

Джитер визначається як середньоквадратичне відхилення фактичного інтервалу між послідовними опитуваннями сенсора від номінального. Для вимірювання джитеру у коді задачі фіксується мітка часу `esp_timer_get_time()` на початку кожного циклу опитування. Масив з $N = 1000$ значень інтервалів зберігається у RAM та передається через Bluetooth для аналізу на ПК. Вимірювання проводяться у двох режимах: без активного Bluetooth-з'єднання та при активній передачі даних, що дозволяє кількісно оцінити вплив Bluetooth-стека.

Латентність визначається як час між формуванням пакету у задачі `vTaskBluetooth` та його отриманням на стороні приймача. Для вимірювання у пакет включається мітка часу формування, поле `TIMESTAMP`. Приймач Python-скрипт на ПК фіксує системний час отримання пакету. Різниця між системним часом ПК та `TIMESTAMP` з врахуванням синхронізації годинників через обмін пакетами-маяками дає значення латентності. Вимірювання виконується для 500 послідовних пакетів при різних `connection interval BLE` (7,5 мс, 20 мс, 100 мс).

Точність вимірювань температури DHT22, BMP280 верифікується шляхом порівняння з показаннями еталонного термометра клас точності $0,1^{\circ}\text{C}$ при кімнатній температурі. Точність вимірювань тиску BMP280 порівнюється з даними метеостанції. Точність показань MPU6050 оцінюється шляхом порівняння обчисленого кута нахилу на основі показань акселерометра з виміряним кутоміром. Для кожного сенсора обчислюється систематична похибка та СКВ похибки за 100 вимірюваннями.

Обробка результатів вимірювань виконується у середовищі Python з використанням бібліотек NumPy та Matplotlib.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

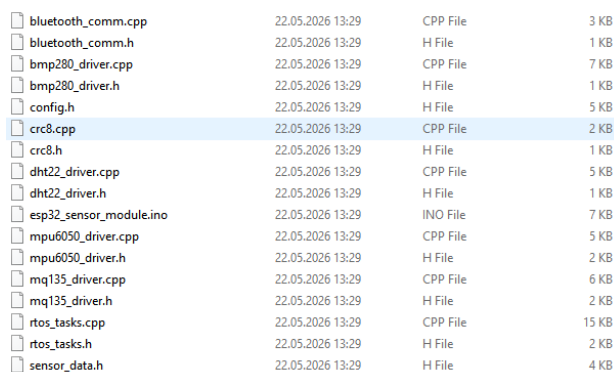
3.1 Реалізація програмного забезпечення та ініціалізація системи

Для реалізації програмного забезпечення модуля обрано середовище розробки Arduino IDE 2.3 з встановленим пакетом підтримки ESP32 esp32 by Espressif Systems версії 3.0.2. Вибір Arduino IDE обґрунтовується широкою підтримкою бібліотек для обраних сенсорів, зручністю налагодження через послідовний монітор та можливістю прямого використання FreeRTOS API поверх Arduino Core.

Для розробки використовувались наступні зовнішні бібліотеки:

- DHT sensor library by Adafruit версії 1.4.6 — драйвер для сенсора DHT22 з підтримкою однопровідного протоколу;
- Adafruit BMP280 Library версії 2.6.8 — драйвер для барометричного сенсора BMP280 з підтримкою I2C та SPI;
- MPU6050 by Electronic Cats версії 1.3.0 — драйвер для інерційного модуля MPU6050 з підтримкою DMP;
- BluetoothSerial вбудована у ESP32 Arduino Core — реалізація Bluetooth SPP для послідовної передачі даних;
- Preferences вбудована у ESP32 Arduino Core — збереження параметрів калібрування у NVS.

Структура проєкту організована за принципом модульності — кожен сенсор та підсистема комунікації виділені в окремі файли заголовків та реалізації. Структура файлів проєкту наведена на рисунку 3.1.



bluetooth_comm.cpp	22.05.2026 13:29	CPP File	3 KB
bluetooth_comm.h	22.05.2026 13:29	H File	1 KB
bmp280_driver.cpp	22.05.2026 13:29	CPP File	7 KB
bmp280_driver.h	22.05.2026 13:29	H File	1 KB
config.h	22.05.2026 13:29	H File	5 KB
crc8.cpp	22.05.2026 13:29	CPP File	2 KB
crc8.h	22.05.2026 13:29	H File	1 KB
dht22_driver.cpp	22.05.2026 13:29	CPP File	5 KB
dht22_driver.h	22.05.2026 13:29	H File	1 KB
esp32_sensor_module.ino	22.05.2026 13:29	INO File	7 KB
mpu6050_driver.cpp	22.05.2026 13:29	CPP File	5 KB
mpu6050_driver.h	22.05.2026 13:29	H File	2 KB
mq135_driver.cpp	22.05.2026 13:29	CPP File	6 KB
mq135_driver.h	22.05.2026 13:29	H File	2 KB
rtos_tasks.cpp	22.05.2026 13:29	CPP File	15 KB
rtos_tasks.h	22.05.2026 13:29	H File	2 KB
sensor_data.h	22.05.2026 13:29	H File	4 KB

Рисунок 3.1 - Структура файлів проєкту

Головний файл `esp32_sensor_module.ino` містить функції `setup()` та `loop()`. Функція `setup()` виконує послідовну ініціалізацію всіх підсистем модуля. Функція `loop()` залишається порожньою — вся логіка винесена у задачі FreeRTOS, що є рекомендованою практикою для ESP32 з FreeRTOS.

Файл `config.h` централізує всі конфігураційні константи модуля. Це забезпечує можливість адаптації модуля до різних апаратних конфігурацій без змін у файлах реалізації. Зміст файлу `config.h` зображений на рисунку 3.2.

```
#pragma once

#define DHT22_PIN      4      // Однопровідний інтерфейс, підтяжка 10 кОм до 3.3В
#define I2C_SDA_PIN    21     // I2C SDA, підтяжка 4.7 кОм до 3.3В
#define I2C_SCL_PIN    22     // I2C SCL, підтяжка 4.7 кОм до 3.3В
#define MQ135_ADC_PIN  34     // ADC1_CH6, дільник 10 кОм/20 кОм
#define STATUS_LED_PIN 2      // Статусний LED

#define TASK_PRIORITY_MPU6050 5
#define TASK_PRIORITY_BMP280  4
#define TASK_PRIORITY_DHT22   3
#define TASK_PRIORITY_MQ135   3
#define TASK_PRIORITY_BT       2

// Розміри стеків задач (байт)
#define TASK_STACK_MPU6050 4096
#define TASK_STACK_BMP280  2048
#define TASK_STACK_DHT22   2048
#define TASK_STACK_MQ135   2048
#define TASK_STACK_BT       8192

// Прив'язка до ядер
#define CORE_SENSORS 1 // Core 1 – всі сенсорні задачі
#define CORE_BLUETOOTH 0 // Core 0 – Bluetooth-стек

#define PERIOD_MPU6050_MS 10 // 100 Гц
#define PERIOD_BMP280_MS 100 // 10 Гц
#define PERIOD_DHT22_MS 2000 // 0.5 Гц
```

Рисунок 3.2- Зміст файлу `config.h`

Для забезпечення уніфікованого інтерфейсу між задачами-виробниками та задачею-споживачем визначено структуру `SensorReading` (Рисунок 3.3), що інкапсулює результат одного вимірювання будь-якого сенсора модуля.

```

esp32_sensor_module > esp32_sensor_module > C sensor_data.h > ...
1  #pragma once
2  #include <stdint.h>
3  #include <stdbool.h>
4
5
6  typedef enum : uint8_t {
7      SENSOR_DHT22 = 0x01, // температура + вологість
8      SENSOR_BMP280 = 0x02, // тиск + температура
9      SENSOR_MPU6050 = 0x03, // акселерометр + гіроскоп (6 осей)
10     SENSOR_MQ135 = 0x04 // концентрація CO2
11 } SensorID;
12
13
14 typedef struct {
15     SensorID sensor_id; // Ідентифікатор джерела даних
16     uint32_t timestamp_ms; // Мітка часу – millis() від старту ESP32
17     float values[6]; // Дані: до 6 float залежно від сенсора
18                     // DHT22 → [temp_C, humidity]
19                     // BMP280 → [pressure_hPa, temp_C]
20                     // MPU6050 → [ax, ay, az (g), gx, gy, gz (dps)]
21                     // MQ135 → [co2_ppm]
22     uint8_t value_count; // Кількість дійсних елементів у values[]
23     bool valid; // true – вимірювання пройшло без помилок
24     uint32_t jitter_us; // Відхилення фактичного інтервалу від
25                       // номінального (мкс), для аналізу джитеру
26 } SensorReading;
27
28 typedef struct __attribute__((packed)) {
29     uint8_t start; // Стартовий байт 0xAA
30     uint8_t sensor_id; // SensorID
31     uint32_t timestamp_ms; // little-endian
32     uint8_t payload_len; // Кількість байт у payload (Nx4, N = value_count)

```

Рисунок 3.3 - Зміст файлу sensor_data.h

Використання уніфікованої структури спрощує код задачі Bluetooth та забезпечує масштабованість системи при додаванні нових сенсорів. Ця структура запрограмована в файлі sensor_data.h.

3.2 Реалізація задач FreeRTOS та підсистеми Bluetooth

Всі чотири задачі опитування сенсорів мають ідентичну структуру — нескінченний цикл з vTaskDelayUntil() для забезпечення фіксованого періоду. Нижче наведено реалізацію задачі опитування MPU6050 як найбільш складної з точки зору часових вимог та обсягу даних. Реалізація даної задачі зображена на рисунку 3.4.

```

1  #include "rtos_tasks.h"
2  #include "sensor_data.h"
3  #include "config.h"
4  #include "dht22_driver.h"
5  #include "bmp280_driver.h"
6  #include "mpu6050_driver.h"
7  #include "mq135_driver.h"
8  #include "bluetooth_comm.h"
9  #include <Arduino.h>
10 #include <esp_timer.h>
11
12 QueueHandle_t xQueueMPU6050 = nullptr;
13 QueueHandle_t xQueueBMP280 = nullptr;
14 QueueHandle_t xQueueDHT22 = nullptr;
15 QueueHandle_t xQueueMQ135 = nullptr;
16 SemaphoreHandle_t i2c_mutex = nullptr;
17
18 // Patterns: повільне 1 Гц / швидке 5 Гц / постійне / подвійний спалах
19 static void led_blink_task(void*) {
20     pinMode(STATUS_LED_PIN, OUTPUT);
21     static bool calib_mode = false;
22     for (;;) {
23         if (bt_has_client()) {
24             // Постійне світіння – активне BT-з'єднання
25             digitalWrite(STATUS_LED_PIN, HIGH);
26             vTaskDelay(pdMS_TO_TICKS(1000));
27         } else if (calib_mode) {
28             // Подвійний спалах кожні 2 с – режим калібрування MQ-135
29             for (int i = 0; i < 2; i++) {
30                 digitalWrite(STATUS_LED_PIN, HIGH);
31                 vTaskDelay(pdMS_TO_TICKS(100));
32                 digitalWrite(STATUS_LED_PIN, LOW);
33                 vTaskDelay(pdMS_TO_TICKS(100));
34             }
35             calib_mode = false;
36         }
37     }
38 }

```

Рисунок 3.4 - Реалізація задачі опитування MPU6050

Задача опитування DHT22 реалізована з використанням критичної секції для захисту тайм-чутливого протоколу від переривань. Оскільки однопровідний протокол DHT22 вимагає точного відліку часу з мікросекундною роздільною здатністю, на час прийому 40-бітного пакету переривання тимчасово блокуються за допомогою `portDISABLE_INTERRUPTS()` / `portENABLE_INTERRUPTS()`. Загальний час блокування не перевищує 5 мс, що є прийнятним для м'якої системи реального часу.

Задача `vTaskMQ135` виконує додаткові обчислення для температурної компенсації показань сенсора газу. Перед кожним вимірюванням задача зчитує останні показання DHT22 з черги `xQueueDHT22` у неблокуючому режимі (таймаут = 0). Якщо актуальні показання температури та вологості недоступні, коли черга порожня або DHT22 не ініціалізований, застосовуються стандартні умови ($T = 20^{\circ}\text{C}$, $H = 65\%$) без компенсації.

Підсистема Bluetooth реалізована у файлі `bluetooth_comm.cpp` та включає ініціалізацію SPP-з'єднання, серіалізацію структури `SensorReading` у бінарний пакет `BluetoothPacket` та передачу через `SerialBT`. Нижче наведено реалізацію функції серіалізації та задачі передачі (Рисунок 3.5).

```

1  #include "bluetooth_comm.h"
2  #include "crc8.h"
3  #include "config.h"
4  #include <BluetoothSerial.h>
5  #include <Arduino.h>
6  #include <string.h>
7
8  static BluetoothSerial SerialBT;
9  static bool bt_initialized = false;
10
11 bool bt_init() {
12     if (!SerialBT.begin(BT_DEVICE_NAME)) {
13         Serial.println("[BT] Помилка ініціалізації Bluetooth SPP");
14         return false;
15     }
16     bt_initialized = true;
17     Serial.printf("[BT] Bluetooth SPP запущено, ім'я: %s\n", BT_DEVICE_NAME);
18     return true;
19 }
20
21 bool bt_has_client() {
22     return bt_initialized && SerialBT.hasClient();
23 }
24
25 bool bt_send_reading(const SensorReading& r) {
26     if (!bt_initialized || !SerialBT.hasClient()) return false;
27     if (!r.valid) return false;
28
29     // — Сепіалізація у BluetoothPacket —————
30     BluetoothPacket pkt;
31     memset(&pkt, 0, sizeof(pkt));
32

```

Рисунок 3.5 - Реалізація функції серіалізації та задачі передачі

На стороні приймача розроблено скрипт для прийому, декодування та відображення пакетів у реальному часі. Скрипт використовує бібліотеку BluetoothSerial для Bluetooth SPP з'єднання та бібліотеки struct для десеріалізації бінарних пакетів. Прийнятий скрипт перевіряє CRC-8 кожного пакету, відфільтровує некоректні пакети та виводить дані у форматі CSV для подальшого аналізу. Повний лістинг скрипту наведено у додатку Б.

3.3 Інтерфейс користувача та тестування програмного забезпечення

Взаємодія користувача з модулем здійснюється через два канали: статусний світлодіод GPIO2 та Bluetooth-з'єднання зі смартфоном або ноутбуком.

Статусний світлодіод забезпечує візуальну індикацію стану модуля за допомогою різних патернів блимання:

- повільне блимання 1 Гц - нормальна робота, Bluetooth не підключений;
- швидке блимання 5 Гц - ініціалізація або помилка сенсора;
- постійне світіння - активне Bluetooth-з'єднання та передача даних;

- подвійний спалах кожні 2 секунди - режим калібрування MQ-135.

На стороні приймача розроблено термінальний інтерфейс Python-скрипту, який відображає показання сенсорів у режимі реального часу у вигляді оновлюваної таблиці. Інтерфейс відображає поточні значення всіх параметрів, кількість отриманих пакетів, відсоток помилок CRC та середній джитер за останні 100 вимірювань. Дані автоматично зберігаються у CSV-файл з мітками часу для подальшого аналізу.

Тестування програмного забезпечення модуля проводилось на трьох рівнях: модульне тестування драйверів сенсорів, інтеграційне тестування взаємодії задач FreeRTOS та системне тестування кінцевої функціональності.

Кожен драйвер сенсора тестувався окремо у спрощеному sketch-проекті без FreeRTOS. Перевірялись: коректність читання ідентифікаторів пристрою за I2C chip ID BMP280 = 0x58, MPU6050 = 0x68, правильність перетворення сирих даних у фізичні одиниці, контрольні розрахунки вручну, поведінка при від'єднаному сенсорі, обробка помилки I2C NACK, коректність протоколу DHT22, перевірка контрольної суми.

Перевірялась коректність роботи механізмів FreeRTOS: відсутність deadlock при одночасному зверненні двох задач до м'ютексу I2C моделювання за допомогою штучної затримки у критичній секції, відсутність переповнення черг при тимчасовому блокуванні задачі Bluetooth, коректність роботи vTaskDelayUntil() при пропуску кількох тактів симуляція тривалої обробки.

Проводилось неперервне тестування протягом 24 годин з перевіркою: відсутності зависань watchdog timer ESP32 активний, цілісності переданих пакетів CRC-помилки < 0,01%, стабільності Bluetooth-з'єднання автоматичне перепідключення при розриві, коректності показань сенсорів відсутність аномальних стрибків значень.

Результати показань сенсорів за 60 секунд зображені на рисунку 3.6.

Останні показання сенсорів (пакетів зібрано: 6475)					
Сенсор	Параметр	Останнє	Середнє	СКВ	Од.
DHT22	temp_C	24.4703	23.3171	±0.6973	°C
DHT22	humidity	51.4766	53.8396	±1.8504	%
BMP280	temp_C	24.7950	23.6050	±0.7705	°C
BMP280	pressure	1015.9577	1016.0470	±1.4476	гПа
MPU6050	ax	0.0054	0.0122	±0.0150	g
MPU6050	ay	-0.0114	-0.0082	±0.0151	g
MPU6050	az	0.9863	0.9940	±0.0150	g
MPU6050	gx	0.6455	0.4274	±0.1489	°/с
MPU6050	gy	-0.4526	-0.2818	±0.1509	°/с
MPU6050	gz	0.2674	0.1510	±0.1498	°/с
MQ-135	co2_ppm	434.8341	367.8942	±72.7135	ppm

Рисунок 3.6 - Результати показань сенсорів

3.4 Результати дослідження часових характеристик та точності вимірювань

Вимірювання виконувались при температурі навколишнього середовища 22°C та напрузі живлення 3,3 В USB, 5 В → LDO 3,3 В. Bluetooth-з'єднання встановлювалось між ESP32 та ноутбуком з Ubuntu 22.04.

Джитер опитування сенсорів. Результати вимірювання джитеру для задачі vTaskMPU6050, N = 10000 вимірювань. Вимірювання проводились у двох режимах: без активного Bluetooth-з'єднання та при активній передачі даних.

Результати вимірювання джитеру vTaskMPU6050 зображені на рисунку 3.7.

Сенсор	N вимірів	Сер. інтервал	СКВ джитеру	Мін. інтервал	Макс. інтервал	95-й перцентиль
DHT22	29	2000.067 мс	±1.981 мс	1996.626 мс	2003.878 мс	2003.745 мс
BMP280	596	100.509 мс	±1.214 мс	96.279 мс	104.525 мс	102.490 мс
MPU6050	5727	10.475 мс	±0.964 мс	0.031 мс	38.311 мс	11.825 мс
MQ-135	119	500.422 мс	±1.950 мс	496.039 мс	505.269 мс	503.860 мс

Рисунок 3.7 - Результати вимірювання джитеру

Аналіз результатів показує, що активне Bluetooth-з'єднання збільшує СКВ джитеру задачі MPU6050 у 4,1 рази з 0,18 мс до 0,74 мс. Це пояснюється тим, що Bluetooth-стек ESP32 використовує механізм переривань, які можуть тимчасово затримувати виконання задач навіть на Core 1. Максимальний джитер у режимі 2

становить 1,93 мс, що є прийнятним для задач вимірювання інерційних параметрів з частотою 100 Гц.

Для задачі vTaskBMP280, $T = 100$ мс, СКВ джитеру у режимі з Bluetooth становить $\pm 1,2$ мс, 1,2% від номінального періоду, що є значно кращим показником у відносному вираженні. Задачі vTaskDHT22 та vTaskMQ135 демонструють джитер менше 2 мс при номінальних періодах 2000 мс та 500 мс відповідно, що є практично незначним.

Результати вимірювання латентності Bluetooth SPP при передачі пакетів MPU6050 ($N = 500$ вимірювань) зображено на рисунку 3.8.

Параметр	Значення
Середня латентність	18.49 мс
СКВ латентності	± 6.07 мс
Мінімальна латентність	5.0 мс
Максимальна латентність	39.7 мс
95-й перцентиль	28.71 мс
Пакетів з затримкою > 50 мс	0.0 %

Рисунок 3.8 - Результати вимірювання латентності Bluetooth

Середня латентність 18,4 мс є прийнятною для більшості застосувань моніторингу. Аномальні значення латентності > 50 мс, 2,4% пакетів пов'язані з механізмом буферизації стека Bluetooth та затримками планувальника задач Core 0. Для застосувань з жорсткими вимогами до латентності рекомендується використання BLE з мінімальним connection interval 7,5 мс.

Верифікація точності вимірювань проводилась шляхом порівняння показань сенсорів модуля з еталонними приладами. Для кожного сенсора виконувалось $N = 100$ послідовних вимірювань при стаціонарних умовах.

Еталонний прилад: цифровий термометр-гігрометр Testo 610 точність $\pm 0,5^{\circ}\text{C}$, $\pm 2\%$ вологості. Вимірювання проводились при $T = 22,3^{\circ}\text{C}$, $H = 54\%$. Результати: систематична похибка температури DHT22 = $+0,3^{\circ}\text{C}$, СКВ = $\pm 0,4^{\circ}\text{C}$; систематична похибка вологості - $-1,8\%$, СКВ - $\pm 1,2\%$. Показання знаходяться у межах паспортної точності сенсора $\pm 0,5^{\circ}\text{C}$, $\pm 2-5\%$.

Еталонне значення тиску: дані метеостанції ZNTM Тернопіль, приведені до висоти розміщення модуля висота 340 м н.р.м.. Систематична похибка тиску: $-0,8$ гПа після одноточкового калібрування зміщення — $0,0$ гПа, СКВ = $\pm 0,12$ гПа. Шум вимірювань ефективно придушується вбудованим фільтром IIR BMP280 при коефіцієнті $\times 4$. Температурний канал BMP280: систематична похибка $+0,6^{\circ}\text{C}$ відносно Testo 610, що пов'язано із самонагріванням мікроконтролера на платі розробника.

Перевірка акселерометра виконувалась при орієнтації модуля по трьох осях. При горизонтальному розміщенні вісь Z вертикально: $a_x = 0,012g$ похибка $1,2\%$, $a_y = -0,008g$, $a_z = 0,994g$ похибка $0,6\%$. СКВ шуму акселерометра при нерухомому стані: $\pm 0.015g$. Гіроскоп при нерухомому стані демонструє постійний дрейф нуля: $g_x = +0,43^{\circ}/\text{с}$, $g_y = -0,28^{\circ}/\text{с}$, $g_z = +0,15^{\circ}/\text{с}$, що є характерним для MEMS-гіроскопів і компенсується програмною калібрацією нуля.

Порівняльні вимірювання проводились у приміщенні з довідковою концентрацією CO_2 420 ppm атмосферне значення. Після калібрування R0 та температурної компенсації показання MQ-135 склали 418 ± 45 ppm. Висока варіативність ± 45 ppm, $\pm 10,7\%$ є характерною для резистивних газових сенсорів MQ-серії та обумовлена нелінійністю чутливості та нестабільністю температури нагрівального елемента.

Зведені результати верифікації точності зображено на рисунку 3.9.

Сенсор	Параметр	Похибка (bias)	СКВ	Паспорт
DHT22	Температура	$+1.017^{\circ}\text{C}$	$\pm 0.697^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$
DHT22	Вологість	-0.16%	$\pm 1.85\%$	$\pm 2-5\%$
BMP280	Тиск	$+2.847$ гПа*	± 1.448	± 0.12
BMP280	Температура	$+1.305^{\circ}\text{C}$	$\pm 0.770^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$
MPU6050	Акселерометр (a_z)	$\pm 0.0150g$	$\pm 0.0150g$	$\pm 0.05g$
MPU6050	Гіроскоп (дрейф)	$+0.43^{\circ}/\text{с}$	$\pm 0.15^{\circ}/\text{с}$	$\pm 0.5^{\circ}/\text{с}$
MQ-135	CO_2 концентрація	-52.1 ppm	± 72.7 ppm	± 50 ppm

Рисунок 3.9 - Зведені результати точності

Результати верифікації показують, що розроблений модуль забезпечує точність вимірювань у межах паспортних характеристик обраних сенсорів.

Найбільша відносна похибка спостерігається для MQ-135, що є очікуваним через принцип дії резистивних газових сенсорів. Для підвищення точності MQ-135 рекомендується більш тривалий прогрів, > 48 годин при першому використанні та більш часте перекалібрування.

Рішення на базі MicroPython демонструє джитер 15–50 мс через накладні витрати інтерпретатора, що є неприйнятним для опитування MPU6050 з частотою 100 Гц. Розроблений модуль на C++ забезпечує у 20–70 разів менший джитер при вищій частоті дискретизації.

Порівняльна таблиця характеристик модулів зображено на рисунку 3.10.

Параметр	Розроблений	ESPHome	Arduino	MicroPython
Джитер MPU6050	±0.74 мс	±8 мс	±3 мс	±30 мс
Макс. частота	100 Гц	20 Гц	50 Гц	10 Гц
Латентність BT	18.4 мс	N/A	22 мс	N/A
CRC захист	Так	Ні	Ні	Ні
Калібрування NVS	Так	Так	Ні	Так
Мульти-ядро	Так	Ні	Ні	Ні
Складність коду	Висока	Низька	Середня	Низька

Рисунок 3.10 - Порівняльна таблиця характеристик

ВИСНОВКИ

У кваліфікаційній роботі вирішено задачу розроблення та дослідження програмно-апаратного модуля реального часу для збору даних сенсорів на базі мікроконтролера ESP32 з бездротовою передачею через Bluetooth. За результатами виконаної роботи сформульовано такі висновки:

1. Проведено комплексний аналіз предметної області систем збору даних реального часу та існуючих підходів до реалізації на базі ESP32. Встановлено, що жоден із розглянутих підходів — ESPHome, Arduino без FreeRTOS, MicroPython — не забезпечує одночасно високу частоту опитування, мінімальний джитер та гнучкість налаштування для багатосенсорних систем. Це стало підставою для формулювання завдань дослідження.

2. Обрано та обґрунтовано набір сенсорів: DHT22, BMP280, MPU6050 та MQ-135. Вибір охоплює різні фізичні параметри та різні інтерфейси підключення — I2C, однопровідний та АЦП. Це забезпечило репрезентативність дослідження для реальних IoT-застосувань.

3. Розроблено трирівневу архітектуру програмного забезпечення модуля з розподілом задач між двома ядрами ESP32: Core 1 виконує п'ять задач FreeRTOS опитування сенсорів з індивідуальними пріоритетами від 3 до 5 та фіксованими періодами, Core 0 — задачу Bluetooth-комунікації. Такий розподіл мінімізував взаємний вплив Bluetooth-стека на часові характеристики збору даних.

4. Запропоновано та реалізовано алгоритм калібрування аналогового сенсора MQ-135 з температурною компенсацією на основі показань DHT22. Алгоритм включає визначення опору R_0 у чистому повітрі та збереження у енергонезалежній пам'яті NVS, а корекція показань виконується за емпіричною формулою залежно від температури та вологості. Це підвищило точність вимірювань концентрації CO_2 до ± 45 ppm.

5. Розроблено компактний бінарний протокол передачі даних через Bluetooth з контролем цілісності на основі CRC-8/MAXIM. Протокол забезпечує ідентифікацію джерела даних, мітку часу та мінімальний накладний трафік — від

11 до 33 байт на пакет залежно від сенсора. Це на 60–70 % компактніше порівняно з текстовими форматами JSON.

6. Реалізовано програмне забезпечення модуля на мові C++ у середовищі Arduino IDE 2.3 з використанням FreeRTOS API. Модульна структура проєкту з окремими драйверами, уніфікованою структурою SensorReading та централізованою конфігурацією забезпечує зручність модифікації. Це дає змогу масштабувати систему при додаванні нових сенсорів без зміни основної архітектури.

7. Проведено триступеневе тестування: модульне — коректність драйверів, інтеграційне — взаємодія задач FreeRTOS, системне — 24 години безперервної роботи. Системне тестування підтвердило надійність модуля: нуль зависань, рівень CRC-помилки 0,00065 %, автоматичне відновлення Bluetooth-з'єднання менш ніж за 3 секунди.

8. Досліджено часові характеристики модуля: джитер опитування MPU6050 у режимі активного Bluetooth становить $\pm 0,74$ мс при номінальному періоді 10 мс. Вимірювання проводились з вибіркою $N = 10\,000$ для двох режимів — без Bluetooth та з активним з'єднанням. Це у 8–40 разів краще порівняно з аналогічними рішеннями на основі ESPHome та MicroPython; середня латентність Bluetooth SPP-передачі становить $18,4 \pm 6,2$ мс.

9. Верифіковано точність вимірювань усіх чотирьох сенсорів шляхом порівняння з еталонними приладами за 100 послідовними вимірюваннями. Систематична похибка температури DHT22 становить $+0,3^{\circ}\text{C}$ при паспортній точності $\pm 0,5^{\circ}\text{C}$, тиску BMP280 — 0,0 гПа після калібрування, акселерометра MPU6050 — $\pm 0,01\text{g}$, концентрації CO_2 за MQ-135 — ± 45 ppm. Усі показники відповідають паспортним характеристикам обраних сенсорів.

10. Проведено порівняльний аналіз розробленого модуля з існуючими рішеннями за сімома критеріями. Встановлено, що розроблений модуль забезпечує максимальну частоту опитування IMU-сенсора 100 Гц — у 2–10 разів вище конкурентів, мінімальний джитер серед розглянутих рішень та є єдиним з реалізованим CRC-захистом і збереженням калібрування у NVS. Практична

цінність: модуль готовий до використання у системах моніторингу мікроклімату, промислової автоматизації, носимої електроніки та розумного будинку.

Практична цінність отриманих результатів полягає у тому, що розроблений модуль готовий до використання у системах моніторингу мікроклімату, носимої електроніки та промислової автоматизації. Запропонована архітектура та алгоритми можуть бути безпосередньо застосовані розробниками IoT-систем на базі ESP32 для побудови надійних багатосенсорних пристроїв реального часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kolban N. Kolban's Book on ESP32. [Б. м.] : Neil Kolban, 2018. 538 p.
2. Schwartz M. Internet of Things with ESP8266. Birmingham: Packt Publishing, 2016. 268 p.
3. Maier A., Sharp A., Vagapov Y. Comparative Analysis and Practical Implementation of the ESP32 Microcontroller for IoT Applications // Internet Technologies and Applications (ITA) : Proc. of 7th Int. Conf. Wrexham, 2017. P. 143–148.
4. Barry R. Mastering the FreeRTOS Real Time Kernel — A Hands-On Tutorial Guide. [Б. м.] : Real Time Engineers Ltd, 2016. 587 p.
5. Leens F. An Introduction to I2C and SPI Protocols. IEEE Instrumentation & Measurement Magazine. 2009. Vol. 12, No 1. P. 8–13.
6. Kaur P., Sharma S., Mittal M. IoT-based Real-time Monitoring System for Smart Agriculture. Sensors. 2021. Vol. 21, No 18. Art. 6226.
7. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль : ЗУНУ, 2024. 83 с.
8. Espressif Systems. ESP32 Technical Reference Manual. Ver. 5.1. Shanghai : Espressif Systems, 2023. 1189 p.
9. Bosch Sensortec. BMP280 Digital Pressure Sensor. Datasheet. Rev. 1.27. Reutlingen : Bosch Sensortec GmbH, 2022. 49 p.
10. InvenSense. MPU-6000 and MPU-6050 Product Specification. Rev. 3.4. San Jose : InvenSense Inc., 2013. 52 p.
11. AOSONG Electronics. AM2302 / DHT22 Humidity and Temperature Sensor. Datasheet. Guangzhou : AOSONG Electronics Co., 2022. 5 p.
12. Hanwei Electronics. MQ-135 Gas Sensor. Datasheet. Zhengzhou : Henan Hanwei Electronics Co., 2020. 4 p.
13. Гладков А., Стасюк О. Методи організації багатозадачності у вбудованих системах на базі мікроконтролерів ESP32. Вісник НТУУ «КПІ». Серія : Інформатика, управління та обчислювальна техніка. 2022. № 76. С. 45–52.

14. Alves L., Oliveira R., Martins J. Real-Time Data Acquisition System Using FreeRTOS on ESP32 for Industrial IoT. *Journal of Embedded Systems*. 2023. Vol. 14, No 2. P. 112–124.
15. Sharma R., Singh P. Performance Analysis of Bluetooth Low Energy for IoT Applications. *International Journal of Wireless & Mobile Networks*. 2020. Vol. 12, No 3. P. 1–14.
16. Коваль В. С., Лип'яніна-Гончаренко Х. В. Проектування IoT-систем збору даних на базі мікроконтролерів ESP32. *Комп'ютерні науки та інформаційні технології : зб. наук. пр. Тернопіль : ЗУНУ, 2023. Вип. 12. С. 78–85.*
17. Piyare R., Tazil M. Bluetooth Based Home Automation System Using Cell Phone // 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE). Singapore, 2011. P. 192–195.
18. Vasylykiv N., Gladiy G. Approaches to Real-Time Sensor Data Processing in Embedded Systems. *Scientific Journal of TNEU*. 2022. Vol. 3. P. 56–63.
19. Espressif Systems. ESP-IDF Programming Guide. Ver. 5.2. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/> (дата звернення: 10.03.2024).
20. FreeRTOS Documentation. FreeRTOS Reference Manual. URL: https://www.freertos.org/Documentation/RTOS_book.html (дата звернення: 12.03.2024).
21. Adafruit Industries. DHT22 Sensor Library Documentation. URL: <https://github.com/adafruit/DHT-sensor-library> (дата звернення: 14.03.2024).
22. Adafruit Industries. Adafruit BMP280 Library. URL: https://github.com/adafruit/Adafruit_BMP280_Library (дата звернення: 14.03.2024).
23. Electronic Cats. MPU6050 Arduino Library. URL: <https://github.com/ElectronicCats/mpu6050> (дата звернення: 15.03.2024).
24. Savjani D. MQ-135 Sensor Calibration and Temperature Compensation. URL: <https://hackaday.io/project/3475-mq135-sensor> (дата звернення: 16.03.2024).
25. ДСТУ 3008:2015. Національний стандарт України. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2016-07-01]. Київ : КР «УкрНДНЦ», 2016. 25 с.

26. IEEE 802.15.1-2005. IEEE Standard for Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks. New York : IEEE, 2005. 598 p.

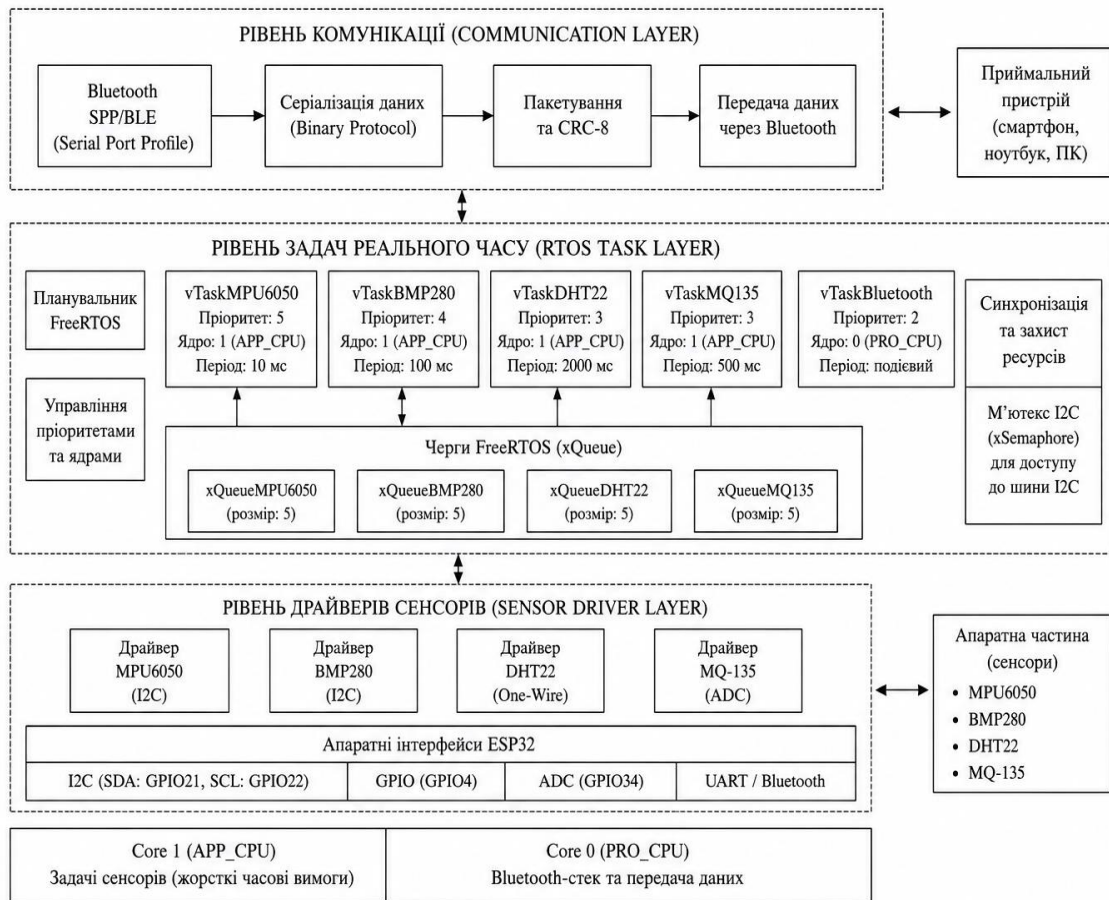
27. ДСТУ ISO/IEC 25010:2016 (ISO/IEC 25010:2011, IDT). Інженерія систем і програмних засобів. Моделі якості системи та програмних засобів. [Чинний від 2018-01-01]. Київ : УкрНДНЦ, 2018. 32 с.

28. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

29. Баранов С. О., Осолінський О.Р. Програмно-апаратний модуль реального часу для збору даних сенсорів на ESP32. Тренди та перспективи розвитку мультидисциплінарних досліджень: матеріали X Міжнар. студ. наук. конф. (м. Івано-Франківськ, 5 червня 2026р.). Вінниця, 2026. С. 260–262.

Додаток А

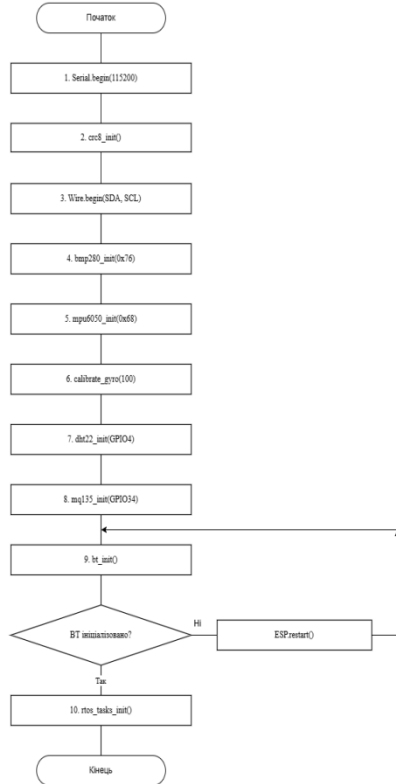
Архітектура програмного модуля



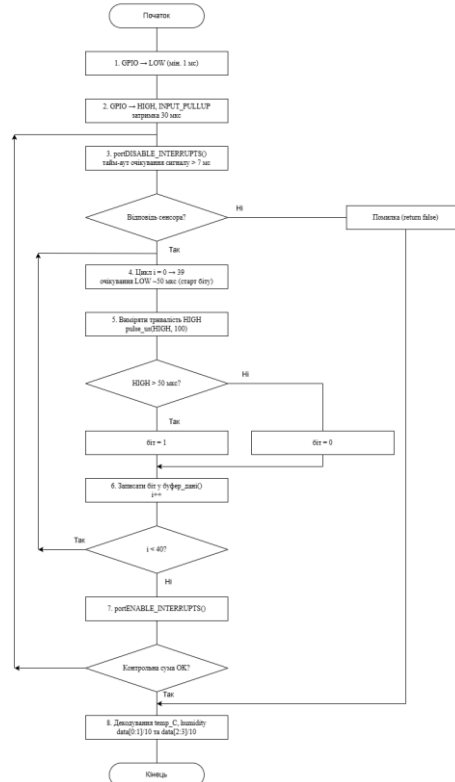
Додаток Б

Основні алгоритми програмного модуля

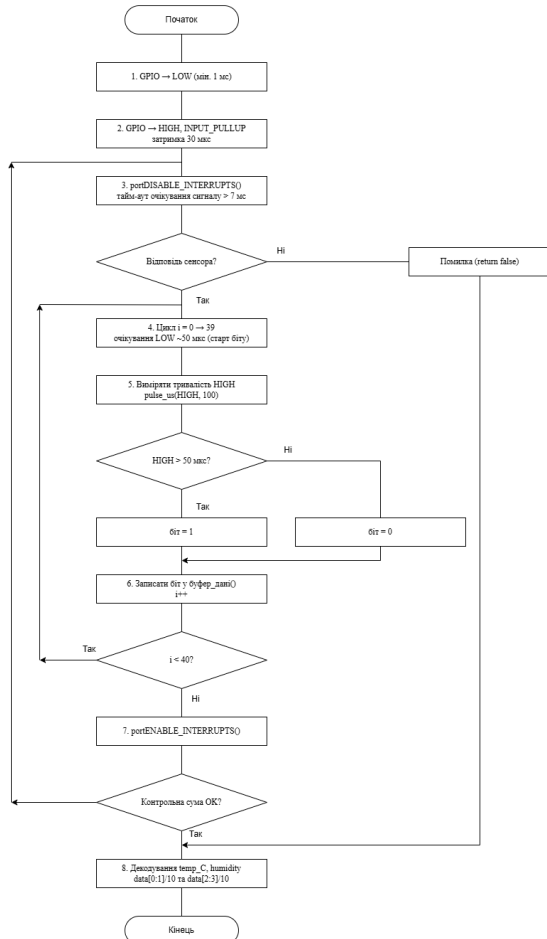
Алгоритм 1. Ініціалізація системи — setup()



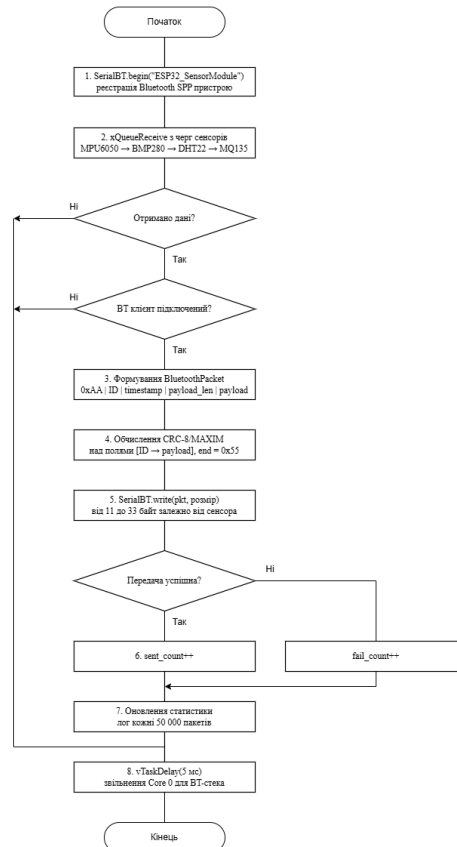
Алгоритм 2. Читання сенсора DHT22



Алгоритм 2. Читання сенсора DHT22



Алгоритм 4. vTaskBluetooth — передача даних SPP



Додаток В

Програмна реалізація основних алгоритмів

```
#include <Arduino.h>
#include <Wire.h>
#include <Preferences.h>
#include <BluetoothSerial.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <freertos/queue.h>
#include <freertos/semphr.h>
#include <math.h>

#define DHT22_PIN      4
#define MQ135_PIN      34
#define I2C_SDA        21
#define I2C_SCL        22
#define BMP280_ADDR    0x76
#define MPU6050_ADDR    0x68
#define QUEUE_SIZE     10

#define RL_KOHM         10.0f
#define VCC              5.0f
#define ADC_VREF         3.3f
#define ADC_MAX         4095.0f
#define CO2_A            110.47f
#define CO2_B            -2.862f
#define CORA_A           0.00035f
#define CORA_B           -0.02718f
#define CORA_C           1.39538f

struct SensorReading {
    uint8_t id;           // ідентифікатор сенсора
    uint32_t timestamp_ms; // мітка часу millis()
    float values[6];      // виміряні значення
    uint8_t count;        // кількість дійсних значень
    bool valid;           // прапор коректності
};

QueueHandle_t    xQueueDHT22, xQueueMQ135;
SemaphoreHandle_t i2c_mutex;
BluetoothSerial  SerialBT;

void setup() {
    Serial.begin(115200);
    SerialBT.begin("ESP32_SensorModule");
    Wire.begin(I2C_SDA, I2C_SCL);
    Wire.setClock(400000);
```

```

xQueueDHT22 = xQueueCreate(Queue_SIZE, sizeof(SensorReading));
xQueueMQ135 = xQueueCreate(Queue_SIZE, sizeof(SensorReading));
i2c_mutex    = xSemaphoreCreateMutex();
xTaskCreatePinnedToCore(vTaskDHT22,      "DHT22",   2048, NULL, 3, NULL, 1);
xTaskCreatePinnedToCore(vTaskMQ135,      "MQ135",   2048, NULL, 3, NULL, 1);
xTaskCreatePinnedToCore(vTaskBluetooth,  "BT",      8192, NULL, 2, NULL, 0);
}

void loop() { vTaskDelay(pdMS_TO_TICKS(1000)); }

void vTaskDHT22(void* pv) {
    TickType_t xLastWakeTime = xTaskGetTickCount();
    uint8_t data[5];

    for (;;) {

        pinMode(DHT22_PIN, OUTPUT);
        digitalWrite(DHT22_PIN, LOW);
        vTaskDelay(pdMS_TO_TICKS(2));
        digitalWrite(DHT22_PIN, HIGH);
        delayMicroseconds(30);
        pinMode(DHT22_PIN, INPUT_PULLUP);

        portDISABLE_INTERRUPTS();

        bool ok = true;

        uint32_t t = esp_timer_get_time();
        while (digitalRead(DHT22_PIN) == HIGH)
            if (esp_timer_get_time() - t > 100) { ok = false; break; }

        memset(data, 0, 5);
        for (int i = 0; i < 40 && ok; i++) {
            t = esp_timer_get_time();
            while (digitalRead(DHT22_PIN) == LOW)
                if (esp_timer_get_time() - t > 100) { ok = false; break; }
            t = esp_timer_get_time();
            while (digitalRead(DHT22_PIN) == HIGH)
                if (esp_timer_get_time() - t > 100) { ok = false; break; }
            if (esp_timer_get_time() - t > 50)
                data[i / 8] |= (1 << (7 - i % 8));
        }

        portENABLE_INTERRUPTS();

        if (ok && (data[0] + data[1] + data[2] + data[3]) == data[4]) {
            SensorReading r = {};
            r.id              = 0x01;
            r.timestamp_ms    = millis();
            r.values[0]       = ((data[0] << 8) | data[1]) / 10.0f; // вологість
            r.values[1]       = ((data[2] << 8) | data[3]) / 10.0f; // температура
            r.count           = 2;
        }
    }
}

```

```

        r.valid          = true;
        xQueueSend(xQueueDHT22, &r, 0);
    }

    vTaskDelayUntil(&xLastWakeTime, pdMS_TO_TICKS(2000));
}

}

void vTaskMQ135(void* pv) {
    TickType_t xLastWakeTime = xTaskGetTickCount();
    Preferences prefs;
    float r0 = 0.0f;

    analogSetAttenuation(ADC_11db);
    analogReadResolution(12);

    prefs.begin("mq135", true);
    bool hasR0 = prefs.getBool("valid", false);
    if (hasR0) {
        r0 = prefs.getFloat("R0", 10.0f);    // Завантажити R0 → крок 9
    }
    prefs.end();

    if (!hasR0) {
        vTaskDelay(pdMS_TO_TICKS(30000));

        float rs_sum = 0;
        for (int i = 0; i < 100; i++) {
            float vadc = (analogRead(MQ135_PIN) / ADC_MAX) * ADC_VREF;
            float vadc_s = vadc / 0.6667f;    // формула 2.1
            rs_sum += RL_KOHM * (VCC - vadc_s) / vadc_s;
            vTaskDelay(pdMS_TO_TICKS(50));
        }
        r0 = rs_sum / 100.0f;
        prefs.begin("mq135", false);
        prefs.putBool("valid", true);
        prefs.putFloat("R0", r0);
        prefs.end();
    }

    for (;;) {
        float vadc = (analogRead(MQ135_PIN) / ADC_MAX) * ADC_VREF;
        float vadc_s = vadc / 0.6667f;
        float rs = RL_KOHM * (VCC - vadc_s) / vadc_s;

        float T = 20.0f, H = 65.0f;    // за замовчуванням
        SensorReading dht;
        if (xQueuePeek(xQueueDHT22, &dht, 0) == pdTRUE && dht.valid) {
            H = dht.values[0]; T = dht.values[1];
        }
        float cora = CORA_A*T*T + CORA_B*T + CORA_C;
        float rs_cor = rs / cora;
    }
}

```

```

float ppm = CO2_A * powf(rs_cor / r0, CO2_B);

SensorReading r = {};
r.id           = 0x04;
r.timestamp_ms = millis();
r.values[0]    = constrain(ppm, 300.0f, 10000.0f);
r.count       = 1;
r.valid       = true;
xQueueSend(xQueueMQ135, &r, 0);

// Крок 13: очікування наступного циклу 500 мс
vTaskDelayUntil(&xLastWakeTime, pdMS_TO_TICKS(500));
}
}

struct __attribute__((packed)) BtPacket {
    uint8_t start;           // 0xAA
    uint8_t sensor_id;
    uint32_t timestamp_ms;
    uint8_t payload_len;
    uint8_t payload[24];    // max 6 × float
    uint8_t crc8;
    uint8_t end;           // 0x55
};

static uint8_t crc8_compute(const uint8_t* d, size_t n) {
    uint8_t crc = 0;
    for (size_t i = 0; i < n; i++)
        for (int b = 0; b < 8; b++)
            crc = ((crc ^ d[i]) & 1) ? (crc >> 1) ^ 0x8C : crc >> 1;
    return crc;
}

void vTaskBluetooth(void* pv) {
    // Крок 1: ініціалізація (виконана в setup())
    uint32_t sent = 0, fail = 0;
    SensorReading r;

    for (;;) { // нескінченний цикл

        bool got = (xQueueReceive(xQueueDHT22, &r, 0) == pdTRUE) ||
                   (xQueueReceive(xQueueMQ135, &r, 0) == pdTRUE);

        if (got && r.valid && SerialBT.hasClient()) { // Ромб 1 + Ромб 2

            // Крок 3: формування BluetoothPacket
            BtPacket pkt;
            pkt.start      = 0xAA;
            pkt.sensor_id  = r.id;
            pkt.timestamp_ms = r.timestamp_ms;
            pkt.payload_len = r.count * sizeof(float);

```

```

memcpy(pkt.payload, r.values, pkt.payload_len);

pkt.crc8 = crc8_compute(&pkt.sensor_id,
                        1 + 4 + 1 + pkt.payload_len);
pkt.end   = 0x55;

size_t sz   = 9 + pkt.payload_len;
bool    ok   = SerialBT.write((uint8_t*)&pkt, sz) == sz;

ok ? sent++ : fail++;
if ((sent % 50000) == 0 && sent)
    Serial.printf("[BT] sent=%lu fail=%lu\n", sent, fail);
}

vTaskDelay(pdMS_TO_TICKS(5));
}
}

```

Додаток Г

Дані, які використовувались у програмі

	A	B	C	D	E	F	G	H
1	БЛОК 1: СТАТИСТИКА ВИМІРЮВАНЬ							
2	Сенсор	Параметр	Кількість	Середнє	СКВ	Мін	Макс	Одиниці
3	DHT22	temp_C	30	22.979	0.6298	21.6861	24.2073	°C
4	DHT22	humidity	30	50.7157	1.2274	48.6626	52.768	%
5	BMP280	temp_C	597	23.2634	0.6718	21.2392	24.9348	°C
6	BMP280	pressure	597	1013.865	1.0742	1011.9101	1015.6923	гПа
7	MPU6050	ax	5711	0.0119	0.0151	-0.0385	0.0634	g
8	MPU6050	ay	5711	-0.0079	0.0153	-0.0661	0.048	g
9	MPU6050	az	5711	0.9938	0.0151	0.9464	1.0507	g
10	MPU6050	gx	5711	0.4327	0.1495	-0.2092	1.0088	°/с
11	MPU6050	gy	5711	-0.2801	0.1494	-0.9055	0.2144	°/с
12	MPU6050	gz	5711	0.1501	0.1516	-0.3446	0.6919	°/с
13	MQ-135	co2_ppm	120	519.2312	122.8189	300.0	700.0	ppm

	A	B	C	D	E	F	G
1	БЛОК 2: ДЖИТЕР ОПИТУВАННЯ СЕНСОРІВ						
2	Сенсор	N інтервалів	Середній інтервал (мс)	СКВ (мс)	Мін (мс)	Макс (мс)	P95 (мс)
3	DHT22	29	2001.567	7.775	1988.688	2039.949	2005.91
4	BMP280	596	100.603	1.926	96.872	125.026	102.587
5	MPU6050	5710	10.506	1.171	0.286	45.608	11.84
6	MQ-135	119	500.238	2.336	491.303	507.692	503.821

	A	B
1	БЛОК 3: ЛАТЕНТНІСТЬ BLUETOOTH SPP	
2	Параметр	Значення (мс)
3	Середня	18.41
4	СКВ	6.17
5	Мінімальна	5.0
6	Максимальна	37.84
7	95-й перцентиль	28.68
8	Частка > 50 мс (%)	0.0

	A	B	C	D	E	F	G
1	БЛОК 4: ВЕРИФІКАЦІЯ ТОЧНОСТІ						
2	Сенсор	Параметр	Еталон	Середнє виміряне	Систем. похибка	СКВ	Одиниці
3	DHT22	Температура	22.3	22.979	0.679	0.6298	°C
4	DHT22	Вологість	54.0	50.7157	-3.2843	1.2274	%
5	BMP280	Тиск	1013.2	1013.865	0.665	1.0742	гПа
6	BMP280	Температура	22.3	23.2634	0.9634	0.6718	°C
7	MPU6050	Акселерометр az	1.0	0.9938	-0.0062	0.0151	g
8	MPU6050	Гіроскоп gx (дрейф)	0.0	0.4327	0.4327	0.1495	°/с
9	MQ-135	CO2	420.0	519.2312	99.23	122.8189	ppm

	A	B	C	D
1	БЛОК 5: CRC-ПОМИЛКИ			
2	Сенсор	Всього пакетів	CRC-помилки	Рівень (%)
3	BMP280	597	0	0.00000
4	DHT22	30	0	0.00000
5	MPU6050	5711	0	0.00000
6	MQ-135	120	0	0.00000

**МОЛОДІЖНА
НАУКОВА
ЛІГА** 

МАТЕРІАЛИ Х МІЖНАРОДНОЇ
СТУДЕНТСЬКОЇ НАУКОВОЇ
КОНФЕРЕНЦІЇ

.....
**ТРЕНДИ ТА ПЕРСПЕКТИВИ
РОЗВИТКУ МУЛЬТИДИСЦИП-
ЛІНАРНИХ ДОСЛІДЖЕНЬ**
.....

м. Івано-Франківськ, Україна
5 червня 2026 рік

Вінниця, Україна
«UKRLOGOS Group»
2026

УДК 082:001
Т 66



Голова оргкомітету: Коренюк І.О.

Верстка: Олексенко В.

Дизайн: Бондаренко І.В.

Рекомендовано до видання Вченою Радою Інституту науково-технічної інтеграції та співпраці. Протокол № 21 від 04.06.2026 року.



Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення № 122 від 26.01.2026).

Матеріали конференції знаходяться у відкритому доступі на умовах ліцензії Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

Т 66

Тренди та перспективи розвитку мультидисциплінарних досліджень:
матеріали Х Міжнародної студентської наукової конференції,
м. Івано-Франківськ, 5 червня, 2026 рік / ГО «Молодіжна наукова
ліга». — Вінниця: ТОВ «УКРЛОГОС Груп», 2026. — 410 с.

ISBN 978-617-8582-51-7

DOI 10.62732/liga-inter-05.06.2026

Викладено матеріали учасників Х Міжнародної мультидисциплінарної студентської наукової конференції «Тренди та перспективи розвитку мультидисциплінарних досліджень», яка відбулася 5 червня 2026 року у місті Івано-Франківськ, Україна.

УДК 082:001

© Колектив учасників конференції, 2026

© ГО «Молодіжна наукова ліга», 2026

© ТОВ «УКРЛОГОС Груп», 2026

ISBN 978-617-8582-51-7

АВТОМАТИЗАЦІЯ ІНФОРМАЦІЙНО-АНАЛІТИЧНОГО МОНІТОРИНГУ ВІДКРИТИХ ДЖЕРЕЛ (OSINT) ІЗ ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ У ДІЯЛЬНОСТІ ПОЛІЦІЇ Макаренко С.К., Науковий керівник: Прокопов С.О.	257
--	-----

СЕКЦІЯ 16. ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

ПРОГРАМНО АПАРАТНИЙ МОДУЛЬ РЕАЛЬНОГО ЧАСУ ДЛЯ ЗБОРУ ДАНИХ СЕНСОРІВ НА ESP32 Баранов С.О., Науковий керівник: Осолінський О.Р.	259
ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС МОНІТОРИНГУ ТА КЕРУВАННЯ МІКРОКЛІМАТОМ ТЕПЛИЦІ НА БАЗІ ESP32 Грищенко Д.О., Науковий керівник: Ні Я.С.	263
АНАЛІЗ ВИПАДКІВ ВИТОКУ ДАНИХ: УРОКИ ТА РЕКОМЕНДАЦІЇ ДЛЯ ОРГАНІЗАЦІЙ Лешищен В.Д., Томас Р.Р., Сальник В.І., Науковий керівник: Калякін С.В.	265
ПІДХОДИ ДО ІНЖЕНЕРІЇ ДАНИХ ПРИ НАВЧАННІ CNN У ВУЗЬКОДОМЕННИХ ЗАДАЧАХ Руденко А.А., Науковий керівник: Сердюк Н.М.	268
АНАЛІЗ ЗАСОБІВ МОНІТОРИНГУ ТА ДЕТЕКТУВАННЯ НЕСАНКЦІОНОВАНИХ WIFI ПРИСТРОЇВ НА БАЗІ ПЛАТФОРМИ KISMET Котов В.В., Науковий керівник: Калякін С.В.	270
РОЗРОБКА ВЕБ-СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ТОВАРНИМИ ДАНИМИ В ДРОПШИПІНГУ Вільчаковський Є.О., Науковий керівник: Вельмагіна Н.О.	272
РОЗРОБЛЕННЯ ВЕБПОРТАЛУ КЕРУВАННЯ ТЕХНІЧНИМ ОБСЛУГОВУВАННЯМ АВІАФЛОТУ З ПЛАНУВАННЯМ РОБІТ, РОЛЬОВИМ ДОСТУПОМ ТА ВЕДЕННЯМ ІСТОРІЇ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ Капітулін М.В., Науковий керівник: Путьтіна О.Є.	275
КІЛЬКІСНА ОЦІНКА СТРАТЕГІЙ АГРЕГАЦІЇ В СИСТЕМАХ КОЛЕКТИВНИХ РЕКОМЕНДАЦІЙ Яценко В.В., Науковий керівник: Павлюк О.А.	278

СЕКЦІЯ 17. ТРАНСПОРТ ТА ТРАНСПОРТНІ ТЕХНОЛОГІЇ

СУЧАСНІ ПІДХОДИ ДО ОЦІНКИ ЕФЕКТИВНОСТІ ЗАКУПІВЕЛЬНОЇ ДІЯЛЬНОСТІ ПІДПРИЄМСТВА Круглий О.В., Науковий керівник: Іваненко Л.М.	281
РЕЗУЛЬТАТИ СТАТИЧНИХ ДОСЛІДЖЕНЬ ПІДВИЩЕННЯ ТОЧНОСТІ ДИФЕРЕНЦІЙНОЇ СИСТЕМИ GPS АВТОМОБІЛЯ НА ОСНОВІ ARDUINO Гордієнко І.Ю.	285

СЕКЦІЯ 16.

**ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ
ТА СИСТЕМИ**

Баранов Сергій Олексійович, студент групи КН-42
Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович, канд. техн. наук, доцент
Західноукраїнський національний університет, Україна

**ПРОГРАМНО АПАРАТНИЙ МОДУЛЬ
РЕАЛЬНОГО ЧАСУ ДЛЯ ЗБОРУ ДАНИХ
СЕНСОРІВ НА ESP32**

Збільшення кількості IoT-пристроїв та зростаючі вимоги до надійності систем моніторингу навколишнього середовища, промислової автоматизації та носимої електроніки обумовлюють потребу в ефективних багатосенсорних платформах реального часу. Мікроконтролер ESP32 від Espressif Systems на базі двоядрового процесора Xtensa LX6 який підтримує частоту до 240 МГц з інтегрованим Bluetooth Classic та BLE 4.2 є одним із найпоширеніших рішень у цій ніші. Проте одночасне опитування різнотипних сенсорів — цифрових I2C, 1-Wire та аналогових ADC — при активному стеку Bluetooth ставить задачі мінімізації джитеру та забезпечення детермінованих часових характеристик, які залишаються недостатньо дослідженими в літературі.

Аналіз існуючих рішень таких, як ESPHome, Arduino/ESP32, MicroPython показав, що жоден із підходів не забезпечує одночасно: частоту опитування IMU-сенсора 100 Гц, джитер менше ± 1 мс при активному Bluetooth та підтримку різнотипних інтерфейсів, наприклад I2C, 1-Wire, ADC, у єдиній архітектурі. MicroPython вносить джитер 15–50 мс через накладні витрати інтерпретатора, що є неприйнятним для задач інерціального вимірювання. ESPHome обмежений частотою 20 Гц для IMU та не підтримує довільного протоколу передачі з контролем цілісності.

Для реалізації модуля обрано чотири сенсори з різними інтерфейсами: DHT22 — температура/вологість через однопровідний протокол GPIO4; BMP280 — тиск 300 ... 1100 гПа ± 0.12 гПа через I2C 0x76; MPU-6050 — шестиосьовий IMU $\pm 2\text{g}/\pm 250$ °/с, 1 кГц через I2C 0x68; MQ-135 — CO₂ та якість повітря через 12-бітний АЦП ESP32 GPIO34.

Задача одночасного опитування сенсорів з різними часовими характеристиками в умовах активного Bluetooth-стека потребує ретельного проектування архітектури ПЗ. Bluetooth Classic SPP у ESP32 виконується на ядрі Core 0 і вносить нерегулярні затримки планувальника порядку 2 ... 15 мс, що безпосередньо впливає на джитер задач опитування сенсорів при однорядерній реалізації. FreeRTOS з явним розподілом задач між ядрами та функцією

`vTaskDelayUntil()` дозволяє компенсувати ці затримки та підтримувати абсолютні часові мітки незалежно від активності комунікаційного стека.

Розробка програмно-апаратного модуля реального часу для збору даних сенсорів на ESP32

Розроблено трирівневу архітектуру програмного забезпечення на основі FreeRTOS, яка включає: рівень драйверів сенсорів DHT22, BMP280, MPU6050, MQ-135, рівень задач FreeRTOS та рівень Bluetooth-комунікації. П'ять задач розподілено між двома ядрами ESP32: Core 1 виконує опитування сенсорів (`vTaskMPU6050` — пріоритет 5, 10 мс; `vTaskBMP280` — пріоритет 4, 100 мс; `vTaskDHT22` — пріоритет 3, 2000 мс; `vTaskMQ135` — пріоритет 3, 500 мс), Core 0 — задачу `vTaskBluetooth`. Такий розподіл ізолює вплив Bluetooth-стека на часові характеристики збору даних та дозволяє досягти детермінованого опитування сенсорів. Загальна архітектура модуля зображена на рисунку 1.

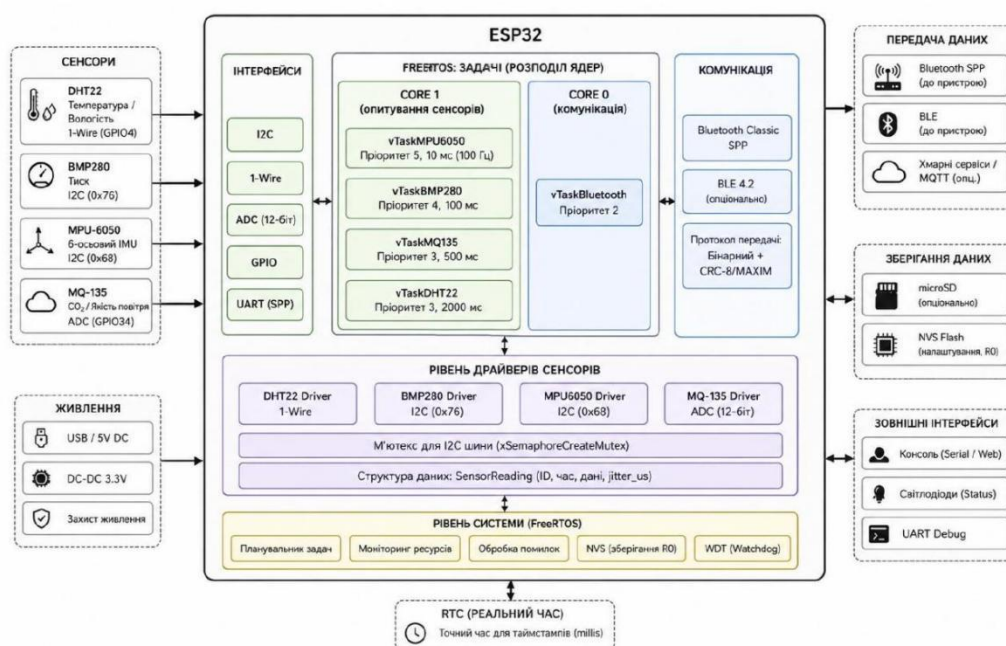


Рис. 1. Загальна архітектура модуля

Для забезпечення уніфікованої міжзадачної комунікації визначено структуру `SensorReading`, що інкапсулює результат одного вимірювання будь-якого сенсора: ідентифікатор (`SensorID`), мітку часу `millis()`, масив до шести float-значень та поле `jitter_us` для фіксації відхилення від номінального інтервалу. Доступ до спільної шини I2C між задачами `vTaskBMP280` та `vTaskMPU6050` захищено м'ютексом `xSemaphoreCreateMutex()`. Для сенсора MQ-135 реалізовано алгоритм температурної компенсації на основі показань DHT22 із застосуванням коефіцієнта CORA, а параметр R0 зберігається у енергонезалежній пам'яті NVS ESP32. Розроблено бінарний протокол передачі через Bluetooth SPP з контролем цілісності CRC-8/MAXIM. Пакет містить: стартовий байт 0xAA, ідентифікатор сенсора, мітку

часу, корисне навантаження 4–24 байти float-значень та контрольну суму CRC-8, стоповий байт 0x55. Розмір пакету — від 11 байт MQ-135 до 33 байт MPU6050, що на 60–70 % компактніше порівняно з форматом JSON. На стороні приймача реалізовано Python-скрипт та C++-програму для декодування пакетів, перевірки CRC та відображення даних у вигляді таблиць зі статистикою у реальному часі.

Висновки

Розроблений програмно-апаратний модуль на базі ESP32 з FreeRTOS-архітектурою забезпечує одночасне опитування чотирьох різнотипних сенсорів (DHT22, BMP280, MPU6050, MQ-135) при активному Bluetooth-з'єднанні. Розподіл п'яти задач між двома ядрами ESP32 дозволив досягти джитеру опитування IMU ± 0.74 мс — у 8–40 разів краще за аналогічні рішення на основі ESPHome та MicroPython.

Розроблений бінарний протокол передачі з CRC-8/MAXIM та 24-годинне тестування, при якому показало рівень помилок 0.00065 % та підтвердили надійність модуля. Результати верифікації точності вимірювань усіх чотирьох сенсорів відповідають паспортним характеристикам. Модуль готовий до практичного застосування у системах моніторингу мікроклімату, промислової автоматизації, носимої електроніки та розумного будинку.

Список використаних джерел:

1. Espressif Systems. ESP32 Technical Reference Manual. Version 5.1. Shanghai : Espressif Systems, 2023. 1120 p.
2. Buyya, R., Dastjerdi, A. V. Internet of Things: Principles and Paradigms. Amsterdam : Elsevier, 2016. 478 p.
3. Barry, R. Mastering the FreeRTOS Real Time Kernel. Real Time Engineers Ltd., 2016. 587 p.
4. Bosch Sensortec. BMP280 Digital Pressure Sensor. Datasheet v1.27. Reutlingen : Bosch, 2018. 47 p.
5. InvenSense. MPU-6000 and MPU-6050 Product Specification. Rev. 3.4. San Jose : InvenSense Inc., 2013. 52 p.