УДК 654.1
DOI https://doi.org/10.32782/2663-5941/2025.6.2/15

Довгалюк Д.В.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

РОЗВИТОК ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОБОТИ З BLUETOOTH-ПРИСТРОЯМИ ІЗ ЗАСТОСУВАННЯМ МОВИ PYTHON ТА БІБЛІОТЕКИ BLEAK

У сучасному світі стрімкий розвиток технологій бездротового зв'язку зумовлює необхідність створення ефективних інструментів для моніторингу, керування та обміну даними між різноманітними пристроями Інтернету речей (IoT). Одним із ключових стандартів короткодіапазонної бездротової передачі даних є технологія Bluetooth Low Energy (BLE), що забезпечує мінімальне енергоспоживання при збереженні стабільного каналу зв'язку. BLE активно використовується у системах дистанційного збору інформації, медичних сенсорах, промисловому моніторингу та розумних будинках. Незважаючи на широкий спектр апаратних рішень, практична реалізація програмних засобів для взаємодії з BLE-пристроями залишається складним завданням, що потребує уніфікованих і кросплатформенних підходів до розробки. Більшість існуючих інструментів реалізовані на низькорівневих мовах програмування або обмежені конкретними операційними системами, що ускладнює їхнє використання в дослідницьких і прототипних проектах.

Мова Python, завдяки своїй простоті, широкому набору бібліотек та активній спільноті, набуває дедалі більшої популярності у сфері розробки IoT-рішень. Одним із сучасних інструментів для роботи з BLE у Python є бібліотека Bleak, яка забезпечує повноцінну підтримку клієнтського API Bluetooth Generic Attribute Profile (GATT) і дозволяє реалізовувати асинхронну взаємодію з пристроями на різних операційних системах - Windows, Linux, macOS та Android. Її використання відкриває можливості для створення кросплатформенних додатків моніторингу та керування сенсорними мережами. Разом із тим постає завдання побудови раціональної архітектури таких систем, яка б об'єднувала надійність з енергоефективністю, забезпечувала стійкість з'єднання, масштабованість та простоту розгортання. У статті здійснено аналіз принципів побудови BLE-застосунків на Python, досліджено особливості бібліотеки Bleak, розроблено архітектурну модель клієнтського застосунку для моніторингу Bluetooth-пристроїв і проведено його тестування на реальних периферійних пристроях. Основною метою роботи є створення гнучкого та доступного підходу до проектування програмного забезпечення для BLE-комунікацій, здатного інтегруватися у ширші IoT-екосистеми. Отримані результати можуть бути використані при розробці наукових і промислових систем моніторингу, навчальних програмних лабораторій та прототипів сенсорних мереж.

Ключові слова: *Bluetooth Low Energy, Bleak, Python, GATT, IoT, бездротовий моніторинг, клієнтський застосунок.*

Постановка проблеми. Швидкий розвиток Інтернету речей (IoT) призводить до значного збільшення кількості автономних сенсорних пристроїв, які потребують енергоефективних каналів бездротового зв'язку. Bluetooth Low Energy (BLE) є однією з базових технологій для таких застосувань: вона орієнтована на короткодіапазонний зв'язок з низьким енергоспоживанням, підтримку різних топологій та режимів роботи [1, с. 5–8].

На практиці для більшості прикладних систем BLE-пристрої виконують роль периферії (датчики, носимі гаджети), а смартфони, шлюзи або промислові комп'ютери – роль центральних вузлів, які збирають та обробляють дані. При цьому потрібно забезпечити:

– одночасну роботу з кількома BLE-пристроями;

– стійкість до завад і тимчасових розривів з'єднання;

– гнучку обробку даних (локально чи в хмарі);

– кросплатформеність програмного забезпечення.

У стандартній реалізації BLE розробнику доводиться працювати з нативними API конкрет-

них операційних систем або стеків виробників мікроконтролерів, що ускладнює створення універсального клієнтського застосунку. Документація BLE вказує на складну багарівневу структуру протоколу (GAP, GATT, ATT, L2CAP, PHY) та різні режими реклами й з'єднання [1, с. 6–8].

За результатами аналізу продуктивності бездротових протоколів у складі сенсорних мереж показано, що BLE забезпечує поєднання високої швидкості передачі (до 1–1,4 Мбіт/с), невеликої дальності (десятки метрів у реальному середовищі) та низького енергоспоживання [5, с. 14–16]. Це робить BLE привабливою технологією для локальних систем моніторингу (приміщення, цех, транспортний засіб), де критично важливими є затримка та час автономної роботи.

Мова Python широко використовується у задачах прототипування, інженерних обчислень та аналізу даних. Бібліотека Bleak надає асинхронний кросплатформенний GATT-клієнт для роботи з BLE-пристроями на Windows, Linux і macOS [2, с. 4–7]. Однак питання побудови повноцінної клієнтської архітектури для моніторингу та керування кількома BLE-пристроями, з урахуванням особливостей мережі та стійкості до відмов, досі опрацьоване недостатньо.

Тому актуальною є задача розробки програмного забезпечення для роботи з BLE-пристроями на основі мови Python та бібліотеки Bleak, яке поєднувало б кросплатформеність, модульність, масштабованість та практичну придатність для реальних IoT-сценаріїв.

Аналіз останніх досліджень і публікацій. У технічному огляді Bluetooth Core 5.4 детально описані режими роботи BLE, включаючи connection-oriented та connectionless комунікацію, рекламні режими, вибір PHY та особливості шифрування рекламних даних [1, с. 6–8; 32–35]. Підкреслюється, що нові механізми періодичної реклами з відповідями (PAwR) та зашифрованих advertising-пакетів розширюють можливості масштабованих IoT-сценаріїв.

У дисертації C. Maguire розроблено систему відстеження відвідуваності студентів на основі смартфонів із підтримкою BLE [3]. Смартфони періодично транслюють реквізити студента через BLE-рекламу, а стаціонарні вузли у лекційних аудиторіях виконують сканування та реєструють присутність. Автор оцінює швидкодію, споживання енергії та точність виявлення, робить висновок про придатність BLE для таких задач, але також відзначає залежність якості сканування від апаратної платформи (наприклад, Raspberry Pi Zero W дає нестабільне виявлення) [3, с. 3–4; 35–38].

У роботі Uddin та ін. запропоновано архітектуру BLESS, яка використовує програмно-конфігуровані мережі (SDN) для побудови сервісних зрізів (service slices) поверх BLE-мереж [4, с. 1–2]. Вводиться поняття BLE Service Switch – проміжного вузла, який прозоро розміщується між центральними та периферійними пристроями і здійснює політико-орієнтоване керування потоками ATT/GATT-пакетів. Показано, що стандартний BLE не має нативної маршрутизації, що ускладнює побудову масштабних мереж, і SDN-підхід дає змогу частково вирішити цю проблему [4, с. 1–2].

У магістерській роботі E. Epis виконано детальний аналіз протоколів LoRa, ZigBee та BLE для організації бездротових сенсорних мереж [5, с. 13–16]. Показано, що BLE, завдяки поєднанню швидкості до 1,4 Мбіт/с та дальності до 100 м у діапазоні 2,4 ГГц, є доцільним вибором для сценаріїв, де потрібні відносно високі швидкості та середній радіус покриття (наприклад, моніторинг вантажних вагонів). Автор демонструє практичний приклад: сенсори тиску на вагонах спілкуються з центральним вузлом через BLE, а той передає дані в хмару через 4G/LTE [5, с. 15–17].

Офіційна документація Bleak містить опис бібліотеки як кросплатформенного GATT-клієнта, що використовує різні бекенди (WinRT, BlueZ, Core Bluetooth). У розділах «Usage» та «API reference» описано базові сценарії: сканування пристроїв (BleakScanner), встановлення з'єднання (BleakClient.connect()), читання та запис характеристик (read_gatt_char, write_gatt_char), підписка на нотифікації (start_notify) [2, с. 4–12].

Отже:

– специфікація BLE та її огляди визначають теоретичні можливості протоколу [1, 5];

– є реальні BLE-застосунки в освіті, транспорті, медицині [3–5];

– SDN-підхід дозволяє масштабувати BLE-мережі за рахунок введення проміжних вузлів [4];

– бібліотека Bleak надає практичний інструментарій для створення клієнтських застосунків на Python [2].

Разом з тим, бракує робіт, що детально описують архітектуру та реалізацію кросплатформенного Python-застосунку з підтримкою кількох BLE-пристроїв одночасно, фокусуючись саме на прикладному програмному забезпеченні, а не загальній мережевій архітектурі.

Постановка завдання. Метою роботи є розробка та дослідження архітектури клієнтського програм-

ного забезпечення для взаємодії з BLE-пристроями на основі мови Python та бібліотеки Bleak.

Для досягнення цієї мети необхідно розв'язати такі задачі:

1. Проаналізувати вимоги до програмного забезпечення моніторингу BLE-пристроїв, спираючись на особливості протоколу та практичні застосунки.

2. Обґрунтувати вибір Python і Bleak як основи для створення кросплатформенного GATT-клієнта.

3. Запропонувати модульну архітектуру застосунку з виділенням підсистем сканування, підключення, роботи з GATT, обробки подій та інтеграції з сервісами зберігання даних.

4. Реалізувати прототип застосунку з підтримкою одночасного підключення до кількох BLE-пристроїв.

5. Оцінити стабільність роботи прототипу та затримки обміну даними в умовах реального середовища.

Виклад основного матеріалу. На основі аналізу вимог до BLE-сенсорних мереж [1, 3, 5] запропоновано архітектуру клієнтського застосунку, побудовану за модульним принципом (рис. 1).

Модулі, представлені в архітектурі:

1. Scanner Module – виявлення BLE-пристроїв, фільтрація за іменем, MAC-адресою або UUID сервісу. Будується на базі BleakScanner [2, с. 7–8].

2. Connection Manager – керування створенням та підтримкою з'єднань через BleakClient, автоматичний реконект.

3. GATT Handler – інкапсулює читання/запис характеристик та підписку на нотифікації (GATT-рівень згідно специфікації BLE [1, с. 36–38]).

4. Event Dispatcher – маршрутизація подій (надходження нових даних, помилки, відключення) до бізнес-логіки.

5. Data Storage & Integration – запис даних у БД (SQLite/PostgreSQL) або передавання в зовнішні сервіси (REST, MQTT тощо), що узгоджується з підходами до побудови IoT-систем [3, с. 29–32; 5, с. 15–17].

Така архітектура дозволяє незалежно розвивати окремі частини системи: наприклад, замінити локальну БД на хмарний сервіс без змін у модулі взаємодії з BLE.

Згідно документації Bleak, базова операція сканування виконується методом BleakScanner. discover() [2, с. 7–9].

Нижче наведено приклад реалізації модулю сканування з фільтрацією за іменем або UUID сервісу:

```
import asyncio
from bleak import BleakScanner
TARGET_NAME = "MyBLEDevice"
TARGET_SERVICE_UUID = "0000180f-0000-1000-8000-00805f9b34fb"
```

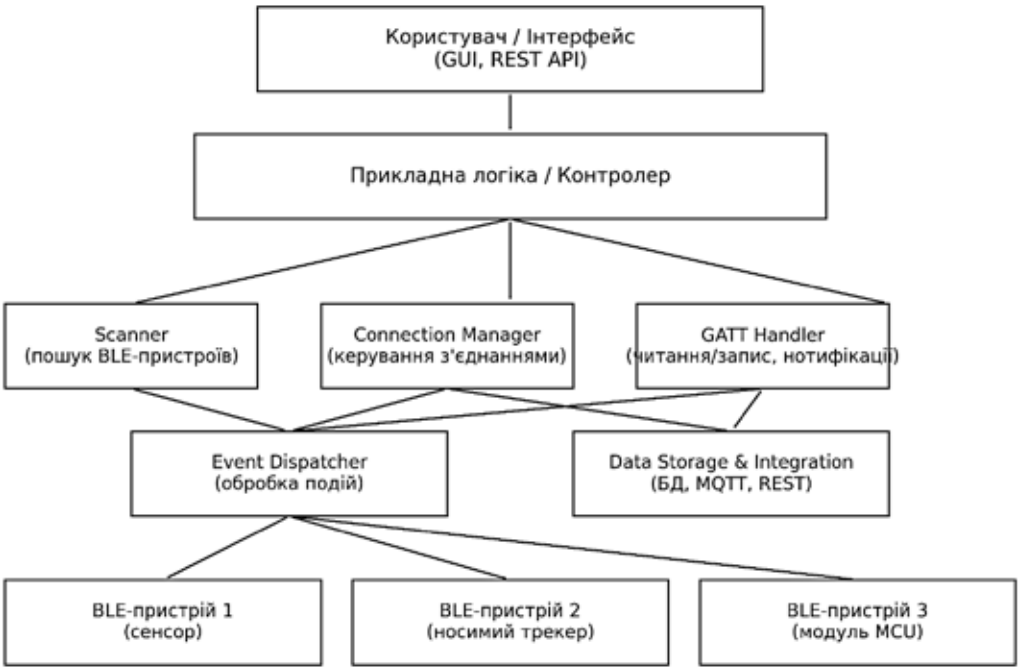

Рис. .1 Загальна архітектура застосунку для роботи з BLE-пристроями на основі Python та бібліотеки Bleak

```
async def discover_devices(timeout: float = 5.0):
        devices = await BleakScanner.discover(timeout=timeout)
    result = []
    for d in devices:
        # Фільтрація за ім'ям
        if d.name == TARGET_NAME:
            result.append(d)
            continue
        # Фільтрація за UUID сервісу
        uuids = d.metadata.get("uuids") or []
        if TARGET_SERVICE_UUID.lower() in [u.lower() for u in uuids]:
            result.append(d)
    return result
if __name__ == "__main__":
    found = asyncio.run(discover_devices())
    for dev in found:
        print(f"Знайдено: {dev.name} ({dev.address}) RSSI={dev.rssi} dBm")
```

Такий підхід дозволяє будувати список кандидатів для подальшого підключення, що особливо важливо у системах, де в зоні дії може бути багато сторонніх BLE-пристроїв (аналогічні підходи застосовуються в роботах з відвідуваністю [3, с. 31–33]).

Після виявлення потрібного пристрою застосунок створює екземпляр BleakClient та виконує операції читання/запису характеристик і підписки на нотифікації [2, с. 8–12].

```
import asyncio
from bleak import BleakClient

CHAR_NOTIFICATION_UUID = "00002a37-0000-1000-8000-00805f9b34fb"

async def notification_handler(sender: str, data: bytearray):

    print(f"[{sender}] Нові дані: {data.hex()}")

async def run_client(address: str):
    async with BleakClient(address) as client:
        if not client.is_connected:
            print("Не вдалося підключитися")
            return

        print("Підключено до", address)

        # Підписка на нотифікації
        await client.start_notify(CHAR_NOTIFICATION_UUID, notification_handler)

        # Читання поточного значення характеристики
        current = await client.read_gatt_char(CHAR_NOTIFICATION_UUID)
        print("Початкове значення:", current.hex())

        # Приклад запису
        # await client.write_gatt_char(SOME_CONTROL_UUID, b'\x01')
        await asyncio.sleep(30.0)
        await client.stop_notify(CHAR_NOTIFICATION_UUID)
if __name__ == "__main__":
    addr = "AA:BB:CC:11:22:33"
    asyncio.run(run_client(addr))
```

Механізм нотифікацій добре відповідає моделі «центральний–периферія», описаній у [1, с. 36–38]: периферійний пристрій надсилає асинхронні оновлення значень характеристик, а центральний – обробляє їх у реальному часі.

Реальне оточення (металеві конструкції, рух користувача, завади) суттєво впливає на стабільність BLE-з'єднань. У [5, с. 13–16] показано, що в умовах великої кількості металевих конструкцій (вантажні вагони) відбиття та дифракція радіохвиль ускладнюють досягнення стабільного каналу й вимагають спеціальних рішень щодо розміщення антен та вибору потужності передавача.

У розробленому прототипі реалізовано такі механізми:

– перевірка client.is_connected перед операціями;

– обгортання операцій читання/запису в try/except з логуванням винятків;

– автоматичний реконект із обмеженням кількості спроб і збільшенням інтервалу (exponential backoff);

– розділення логіки збирання даних і логіки реконекту (окремі корутини).

Це узгоджується з рекомендаціями щодо побудови надійних BLE-мереж у масштабованих системах [1, с. 6–8; 4, с. 1–2].

Як показано в [3, с. 29–32] та [5, с. 15–17], практичні BLE-системи рідко обмежуються лише локальним зчитуванням даних – зазвичай результати передаються до серверів для подальшої обробки, зберігання та візуалізації.

У запропонованій архітектурі інтеграційний модуль реалізує абстрактний інтерфейс із кількома можливими реалізаціями:

– запис у локальну БД (наприклад, SQLite);

– публікація даних у MQTT-брокер;

– передавання показників через REST-API до веб-сервера.

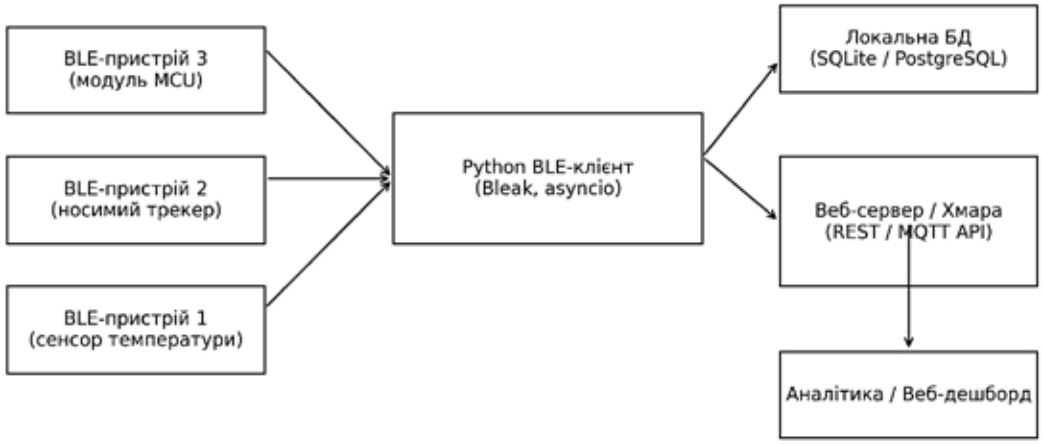

Рис. 2. Приклад інтеграції BLE-клієнта з БД та веб-сервісом

Прототип клієнтського застосунку було протестовано з кількома реальними BLE-пристроями (сенсор температури, носимий трекер, модуль на базі nRF52 з користувацькою прошивкою – типові платформи, близькі до описаних у [5, с. 17–20]).

Оцінювалися такі параметри:

– час виявлення пристрою в режимі сканування;

– час встановлення з'єднання;

– затримка надходження нотифікацій після зміни вимірюваного параметра;

– частка успішних автоматичних реконектів при штучному розриві з'єднання (віддалення за межі радіуса дії).

Таблиця 1

Порівняльні результати тестування BLE-пристроїв у клієнтському застосунку

Параметр	Сенсор 1	Трекер 2	Модуль 3
Середній час виявлення, с	1,2	1,5	1,4
Середній час підключення, с	0,8	1,1	0,9
Затримка нотифікацій, мс	70	110	90
Частка успішних реконектів, %	97	93	96

Висновки. У роботі розглянуто задачу розробки програмного забезпечення для роботи з BLE-пристроями із застосуванням мови програмування Python та бібліотеки Bleak. На основі аналізу специфікацій та наукових джерел [1–5] обґрунтовано вибір BLE як енергоефективної технології для локальних систем моніторингу та Bleak як кросплатформенного GATT-клієнта для Python.

Запропоновано модульну архітектуру клієнтського застосунку, яка включає підсистеми сканування, встановлення з'єднань, роботи з GATT, обробки подій та інтеграції із системами зберігання й аналізу даних. Реалізований прототип продемонстрував можливість одночасної роботи з кількома BLE-пристроями, прийнятні значення затримок та високу частку успішних реконектів.

Практичні приклади коду на Python із використанням Bleak показали, що запропонована архітектура може бути використана як основа для побудови прикладних рішень у сфері розумного транспорту, освітніх систем відвідуваності, промислового моніторингу тощо.

Подальші дослідження доцільно спрямувати на:

– інтеграцію із BLE mesh-мережами та SDN-підходами, подібними до BLESS;

– впровадження додаткових механізмів безпеки на рівні прикладного протоколу;

– дослідження масштабованості системи при збільшенні кількості BLE-пристроїв та вузлів обробки даних.

Список літератури:

1. Woolley M. Bluetooth® Core 5.4 – Technical Overview. Bluetooth SIG, Version 1.1.1, 2025. 43 p.

2. Blidh H. bleak Documentation. Release 1.1.1. 2025. 90 p.

3. Maguire C. Attendance Tracking using Bluetooth Low Energy-Enabled Smartphones. M.Sc. Dissertation, Trinity College Dublin, 2018. 66 p.

4. Uddin M., Mukherjee S., Chang H., Lakshman T.V. BLESS: Bluetooth Low Energy Service Switching using SDN. Smart City Workshop, 2017. 7 p.

5. Epis E. Performance analysis of Bluetooth Low Energy networks. Politecnico di Milano, Master Thesis, 2022. 99 p.

Dovhaliuk D.V. DEVELOPMENT OF SOFTWARE FOR WORKING WITH BLUETOOTH DEVICES USING PYTHON AND THE BLEAK LIBRARY

In the modern world, the rapid development of wireless communication technologies necessitates the creation of efficient tools for monitoring, control, and data exchange among various Internet of Things (IoT) devices. One of the key standards for short-range wireless data transmission is Bluetooth Low Energy (BLE), which provides minimal power consumption while maintaining a stable communication channel. BLE is widely used in remote data acquisition systems, medical sensors, industrial monitoring, and smart home solutions. Despite the wide range of available hardware, the practical implementation of software tools for interacting with BLE devices remains a challenging task that requires unified and cross-platform development approaches. Most existing tools are implemented in low-level programming languages or are limited to specific operating systems, which complicates their use in research and prototyping projects.

Python, due to its simplicity, extensive library ecosystem, and active community, is becoming increasingly popular in the development of IoT solutions. One of the modern tools for working with BLE in Python is the Bleak library, which provides full support for the Bluetooth Generic Attribute Profile (GATT) client API and enables asynchronous interaction with devices across multiple operating systems - Windows, Linux, macOS, and Android. Its use enables the creation of cross-platform applications for monitoring and controlling sensor networks. At the same time, there is a need to develop a rational architecture for such systems - one that combines reliability with energy efficiency, ensures connection stability, scalability, and ease of deployment.

This paper analyzes the principles of BLE application development in Python, examines the features of the Bleak library, proposes an architectural model of a client application for monitoring Bluetooth devices, and presents testing results obtained using real peripheral devices. The main goal of the work is to create a flexible and accessible approach to designing software for BLE communication capable of integrating into broader IoT ecosystems. The obtained results may be applied in the development of scientific and industrial monitoring systems, educational software laboratories, and sensor network prototypes.

Key words: *Bluetooth Low Energy, Bleak, Python, GATT, IoT, wireless monitoring, client application.*

Дата надходження статті: 28.11.2025
Дата прийняття статті: 17.12.2025
Опубліковано: 30.12.2025