https://doi.org/10.31891/2219-9365-2025-84-27
УДК 621.396 + 004.6 + 004.451.4

ГЕРМАН Юрій
Чернівецький національний університет ім. Юрія Федьковича
https://orcid.org/0009-0003-2473-7365
email: herman.yurii@chnu.edu.ua
ВЕРИГА Андрій
Чернівецький національний університет ім. Юрія Федьковича
https://orcid.org/0000-0002-2616-3057
email: a.veryga@chnu.edu.ua

ОФЛАЙН СИСТЕМА ОБРОБКИ ТА ВІЗУАЛІЗАЦІЇ РАДІОСИГНАЛУ

Робота розглядає автономний SDR-сканер для відтворюваних радіотехнічних експериментів у середовищах з обмеженими мережевими та обчислювальними ресурсами. Поставлена задача - забезпечити жорстку причинність та відтворюваність «від приймача до перегляду» без залежності від зовнішніх брокерів і TS-баз: кадр формується в C-вузлі (HackRF + FFTW, N=4096), серіалізується у NDJSON (у форматі «1 подія = 1 рядок») і передається в тонкий бекенд (FastAPI + SQLite/WAL) для одночасного запису та live-стріму до веб-клієнта (SSE/WS). Керування параметрами (Freq/SR/LNA/VGA/Atten) здійснюється через Unix Domain Socket із транзакційною парою txid↔ack, при чому команда застосовується між кадрами. Така організація зберігає часову геометрію (збереження кадрів спирається на часові мітки) й частотні інваріанти (bin_hz), спрощує трасування та дозволяє точне відтворення «як записано». Показано офлайн-систему на Raspberry Pi (ARM64), а також підхід до агрегування інтервалів спостережень (на основі часових проміжків) і керування життєвим циклом збережених проміжків. Запропонована архітектура мінімізує рухомі частини, але залишається здатною до масштабування, що робить її придатною для польових вимірювань, автономного моніторингу, прототипування методів радіоконтролю та розвитку власних рішень.

Ключові слова: SDR, HackRF, embedded, edge-обчислення, автономні системи, Raspberry Pi.

HERMAN Yurii, VERYGA Andrij
Yuri Fedkovych Chernivtsi National University

OFFLINE RADIO SIGNAL PROCESSING AND VISUALIZATION SYSTEM

The paper presents the design and implementation of an autonomous software-defined radio (SDR) scanner intended for reproducible radio engineering experiments in environments with limited network connectivity and constrained computing resources. The primary objective is to ensure strict causality and full reproducibility of the signal processing chain "from receiver to viewing," while eliminating dependencies on external message brokers, time-series databases, or cloud-based infrastructure.

The proposed architecture forms signal frames directly at the capture node using HackRF hardware and FFTW-based spectral processing with a fixed transform size of N = 4096. Each frame is serialized as newline-delimited JSON (NDJSON), following the principle of "one event per line," and transmitted to a lightweight backend implemented with FastAPI and SQLite operating in write-ahead logging (WAL) mode. This backend simultaneously supports persistent storage and real-time streaming to a web-based client via Server-Sent Events (SSE) or WebSocket (WS) interfaces.

Receiver parameter control, including center frequency, sample rate, LNA and VGA gains, and attenuation, is implemented through a Unix Domain Socket interface. A transactional control mechanism based on a txid↔ack handshake guarantees deterministic application of commands strictly between frames, thereby preserving temporal causality. Frame integrity is maintained using precise timestamps, while frequency-domain invariants are ensured through fixed bin spacing (bin_hz), enabling accurate reconstruction and "as recorded" playback of measurement sessions.

The paper also demonstrates a fully offline deployment on a Raspberry Pi (ARM64), confirming the feasibility of autonomous operation in field conditions. In addition, an approach to aggregating observation intervals based on time ranges is introduced, along with lifecycle management of stored intervals for efficient long-term use.

The proposed solution minimizes architectural complexity while remaining scalable and extensible, making it well suited for field measurements, autonomous spectrum monitoring, rapid prototyping of radio control methods, and the development of proprietary SDR-based monitoring and analysis systems.

Keywords: SDR, HackRF, embedded, edge computing, autonomous systems, Raspberry Pi.

Стаття надійшла до редакції / Received 06.11.2025
Прийнята до друку / Accepted 23.11.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

В системах моніторингу, наприклад радіочастотних сигналів, є необхідність передавати результати вимірювань від блоку вимірювань та попереднього оброблення до блоку відображення та керування. Передавання здійснюють, зазвичай, через радіоканал за допомогою трансивера.

Системи подібного типу можна реалізувати з використанням мікрокомп'ютера RaspberryPi. Оскільки він містить модуль Wi-Fi, який можна використовувати для обміну даними між блоками, тому відпадає необхідність в окремому трансивері. Такий підхід надає системі радіомоніторингу здатності працювати автономно.

Більшість відомих платформ потребує або зовнішніх брокерів/кластерів, або вузькоспеціалізованої інфраструктури (TS-сховища, потокові шини), що ускладнює розгортання в якості простих edge-платформ і підвищує ризик втрати причинно-часової цілісності даних. Робота в польових умовах диктує інші вимоги: автономність, короткий шлях даних формату: отримав-обробив-зберіг-відобразив, відсутність залежності від глобальних мереж, чіткі розуміння відповідальності компонентів, передбачувані часові інтервали та можливість відтворити експеримент без спотворення даних.

Особливо, для наукових завдань, важлива формальна відтворюваність:

- часові мітки мають відповідати реальному темпу формування кадрів,
- частотна роздільна здатність - бути однозначно відновлюваною з конфігурації,
- керування - давати транзакційні гарантії послідовної зміни конфігурації,
- збереження - гарантувати відповідність даних їх джерелу.

Складання цих вимог у єдиний мінімалістичний стек є нетривіальною задачею: потрібно узгодити низькорівневий прийом (HackRF), детермінований спектральний конвеєр (FFTW), неблокуюче транспортування і локальний запис (FastAPI + SQLite/WAL), а також легкий браузерний клієнт із live/replay без зовнішніх залежностей.

З огляду на зростання ролі edge-систем у сфері моніторингу спектра радіочастотних сигналів, протидії перешкодам і навчанні, постає завдання продемонструвати, що офлайн-орієнтований підхід - без хмари, зовнішньої інфраструктура та стандартних підходів IoT - може забезпечити потрібну надійність, керованість та відтворюваність, залишаючись простим у розгортанні на Raspberry Pi та аналогічних платформах.

ФОРМУВАННЯ ЦІЛЕЙ СТАТТІ

Мета статті - сформувати та описати автономну архітектуру SDR-сканера, котра забезпечує відтворювану поведінку та керується основними підходами edge-платформ без залежності від зовнішньої інфраструктури.

Для цього потрібно:

1. визначити мінімальний набір компонентів і формальну взаємодію між ними, що гарантують причинність і часову/частотну незмінність (Кожна команда отримує пряму відповідь із її результатом, команди виконуються між отриманням блоків даних, динамічне налаштування параметрів, монотонний годинник для часового відліку);
2. реалізувати тонкий транспортно-сховищний шар із одночасним збереженням і відправкою даних до клієнта;
3. забезпечити компактне подання спектральних кадрів у спрощеному форматі для імпорту/експорту та точного відтворення у подальшому;
4. Делегувати вимогливі до ресурсів операції на клієнт задля спрощення вимогливості самої системи до платформ на якій вона розгорнута
5. Отримані елементи системи повинні підтримувати модифікації незалежно один від одного, при умові притримування спільного формату даних.

ОСНОВНИЙ МАТЕРІАЛ ДОСЛІДЖЕНЬ

Оперативний радіомоніторинг зазвичай потребує зовнішніх ресурсів, потужної техніки та складних процесів – а це вже призводить до проблем в польових умовах та загальної малої надійності системи. Даний підхід орієнтується на вже існуючу інфраструктуру, побудованій мережі та наявність доступу до зовнішніх мереж. Запропонований підхід орієнтується на роботу на “місці” та швидкій обробці даних використовуючи edge(кордонний) підхід до розробки телекомунікаційних систем.

Враховуючи це, було побудовано автономну систему-сканер, котра здатна збирати та відтворювати спектральні дані в умовах обмежених ресурсів та відсутності доступу до зовнішніх мереж.

Мета - імплементувати весь процес обробки сигналу формату:

«приймання → спектральний аналіз → збереження → відображення → експорт»

Для того аби досягти потрібного функціоналу була побудована трьохкомпонентна система:

1. вузол збору з апаратним доступом до HackRF та FFTW для швидкого перетворення Фур’є
2. тонкий бекенд із локальним записом у базу даних
3. веб-інтерфейс із потоковою доставкою та службовим каналом обміну даними.

Постановка задачі обумовлює кілька особливостей.

По-перше, часові характеристики: **bin_hz = sample_rate_hz/N**, мітки часового проміжку - **ts_ms** і «межа кадра», на якій застосовуються зміни конфігурації повинна бути збережена для кожного фрейму даних, оскільки це гарантує відтворюваність збережених даних із демонстрацією обраних налаштувань. По-друге, **всі процеси повинні бути ізольовані один від одного**: керування йде з бекенда до C через UDS (Unix Domain Socket), а дані та відповіді йдуть назад у stdout у форматі NDJSON (рівно один JSON-об’єкт на рядок). Це все дозволяє уникнути взаємного впливу елементів без прямої на те команди, зберігаючи цей обмін незалежним

від каналів вводу та виводу. По-третє, **локальний характер**: вся система повинна бути здатна функціонувати лише на одній платформі, опрацьовувати дані без доступу користувача до неї, взаємодіяти із клієнтом (при його наявності), та зберігати отримані дані у будь-якому випадку.

Сформулювавши задачу було отримано список обмежень на основі яких і були обрані інструменти для побудови системи. **FFTW** дає стабільну продуктивність на ARM при детермінованих параметрах плану, що важливо для сталого темпу кадрів і коректної нормалізації спектра [1]. **HackRF One** - доступний широкосмуговий SDR з передбачуваним API та типовими для вимірювань електромагнітного поля компромісами (смуга, квантування, чутливість) [2]. **Raspberry Pi** - доступна платформа для функціонування подібних систем, щодо якої існує напрацьований блок рішень для обробки сигналів [3]. ЇЇ потужності достатньо для обробки та форматування даних, їх зберігання та подальшої агрегації, що в свою чергу дозволяє тримати всю систему мінімалістичтичною та відкритою до включення зовнішніх інструментів (оскільки ОС базується на ядрі Linux) [4]. Обмін даними проходить у форматі NDJSON що дозволяє спростити обробку та формулювання повідомлень, а кодування цифрових даних зменшує необхідну кількість ресурсів для їх збереження [5]. **SQLite у WAL режимі** - легкий на вставки та стійкий до збоїв журнал, який добре переносить режим «один рядок - одна подія» і відповідає вимогам офлайн-збірки [6]. **SSE/ WebSocket** - стандартні транспортні механізми для обробки потоку даних та двостороннього обміну інформації у вигляді службових повідомлень та постійного каналу для отримання кадрів [7].

Архітектура системи. Поділ системи на три основних компоненти дозволяє використовувати класичні підходи для обміну даними, патерни роботи та надавати можливість до незалежної модифікації кожного елементу без блоку інших. На Рис.1 ілюстровано високорівневу архітектуру за цими блоками та їх функціоналом.

Експортер даних. За отримання та початкову обробку даних відповідає окремий процес, що працює напряму із периферією та здатен реагувати на зовнішні команди. Задля досягнення цього було сформовано алгоритм роботи орієнтований на швидку обробку даних із неблокуючим вводом та виводом. На Рис.2 відображено основну логіку роботи експортера.

Цей шар взаємодіє із HackRF, збирає метадані конфігурації (частота, семплрейт, підсилення), і заздалегідь готує план FFTW (N = 4096) з попередніми оптимізаціями для сталого часу виконання. Приймання ведеться асинхронно, блоки даних обробляються одразу після отримання кадру із периферії, не зупиняючи основний цикл роботи. У результаті, кожні **N** відліків формується повідомлення з інформацією про кадр, списком послідовностей для візуалізації після FFTW, інформацією про налаштування експортера, часом коли це повідомлення було сформоване. Отриману структуру переформатовуємо в NDJSON та надсилаємо в stdout для подальшої обробки.

Рис. 1. Компонентна схема

Паралельно цьому ми також очікуємо команди ззовні, при отриманні яких - кожна транслюється в відповідні зміни у налаштуваннях периферії чи загальної зміни конфігурації, отримана структура вноситься у чергу для виконання.

В основному циклі, на кожне його повторення ми перевіряємо цю чергу і якщо вона не пуста - отримана команда виконується, її результат відправляється в stdout. Незалежно від наявності команди експортер відправляє інформацію про конфігурацію, та стан периферії в stdout на кожне повторення циклу.

Цей процес забезпечує три критичні характеристики. По-перше, часова: темп кадрів дорівнює N/sample_rate_hz, а мітки часу беруться з монотонного годинника на самій платформі. По-друге, частотна: частота включена у кожну подію, порядок двійкових послідовностей є фіксованим, що робить геометрію спектра однозначною. По-третє, причинна: будь-яка команда має txid і породжує відповідний ack, який іде тим самим каналом, що й дані, - отже, всі операції незмішувані і зберігаються в історії даних.

Проміжний шар. Проміжний шар читає NDJSON-потік зі stdout C-процесу, негайно декодує bins_b64 → float32, записує кадри в SQLite у транзакціях і транслює live-потік у браузер. Вставки виконуються у режимі WAL, що добре переносить інтенсивні потоки даних з короткими вставками та дозволяє швидко відновлюватись у випадку збою живлення. При масових операціях (імпорт/видалення інтервалів) запускаються додаткове очищення журналу та зайнятої пам'яті для стримування росту WAL.

На вихід у клієнт бекенд застосовує SSE для масових кадрів спектра (спрощене автопід'єднання, низьке додаткове навантаження на відправку даних, впорядкованість та односторонння направленість) і WebSocket для службових подій: статус, результати команд, нотифікації стану, контроль replay функціоналу та змін конфігурації [8]. Для інтерактивного керування система працює як посередник, та передає команди від користувача на експортер за спільним контрактом, не вносячи змін у суть запитів. Сам сервіс може працювати на локальному інтерфейсі без доступу до зовнішньої мережі, а наявність лімітів дозволяє пріорітизувати збереження даних перед їх відображенням, у разі перевищення навантаження пріоритет надається локальному запису в БД, а потоки обмежуються до цільового fps.

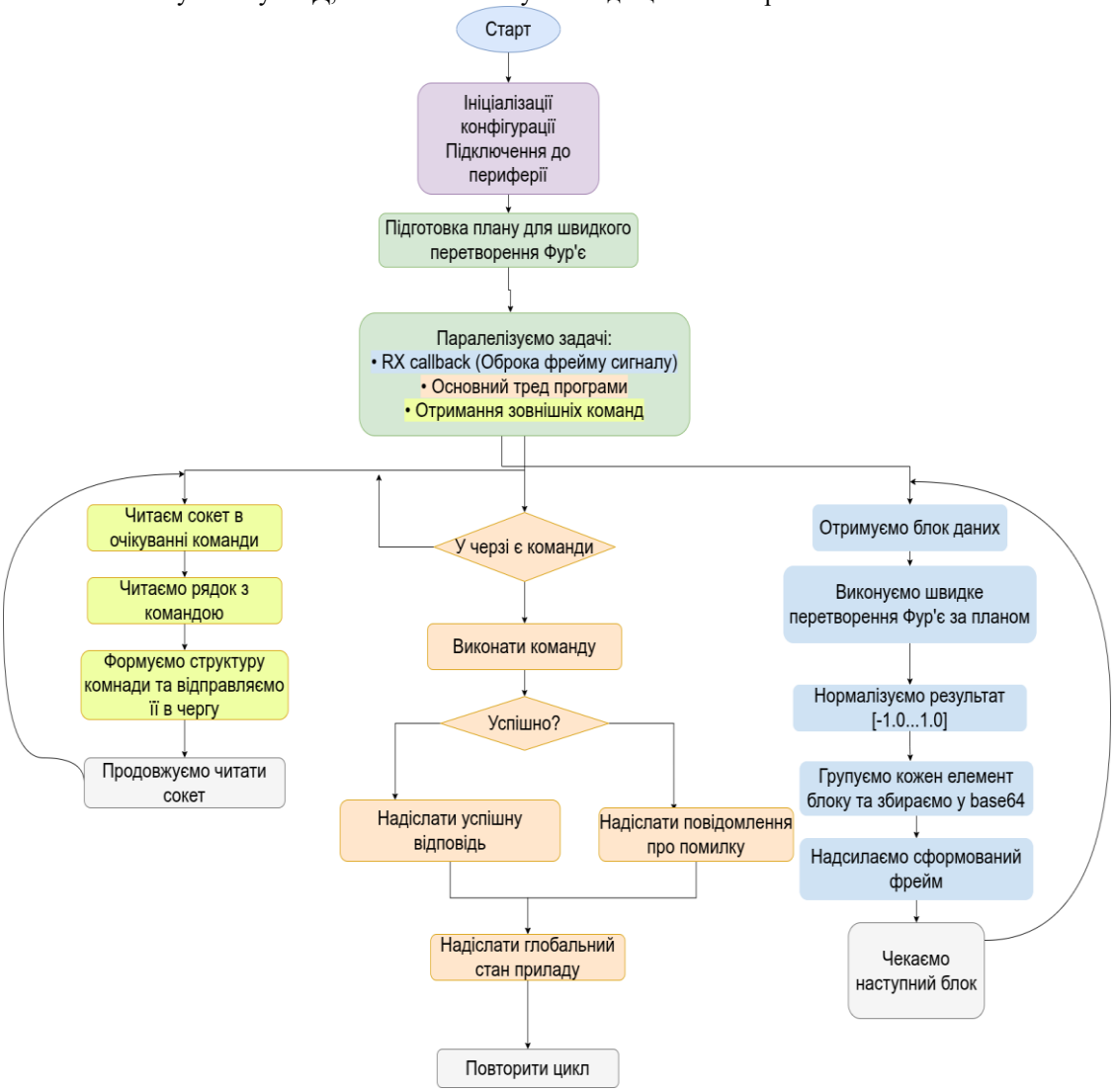

Рис. 2. Процесна діаграма С-вузла

Веб-інтерфейс. Інтерфейс декодує отримані двійкові дані у **WebWorker** [9], щоб не блокувати основний процес, відмальовує графіку у реальному часі (за допомогою WebGL) [10], та надає контроль над параметрами налаштування самого приладу. Це все відображено на Рис.3. де наявна панель керування та відмальований спектр обраного проміжку.

Кожна зміна параметрів супроводжується формуванням ідентифіктора (txid) ; лише після отримання підтвердження у відповідь -інтерфейс позначає стан як застосований. Також підтримується режим **replay** що базується на підході скінченних автоматів та гарантує успішну зміну станів (stop-start-pause), підтримується навігація по часових мітках, керування швидкістю (0.5, 1, 2, 3), якщо сама система працює - при підключенні до неї клієнт одразу отримає інформацію про те які саме параметри було обрано, оскільки кожен отриманий кадр в собі зберігає конфігурацію. Для огляду записів UI показує агреговані **інтервали** на відрузку часу, вікно для агрегації також конфігуроване, що дозволяє контролювати наскільки деталізовано система здатна розбити наявні кадри на часові блоки [11].

У загальному, побудована система надає основний функціонал для роботи із аналізом радіосигналів та підтримує подальшу їх агрегацію за потреби.

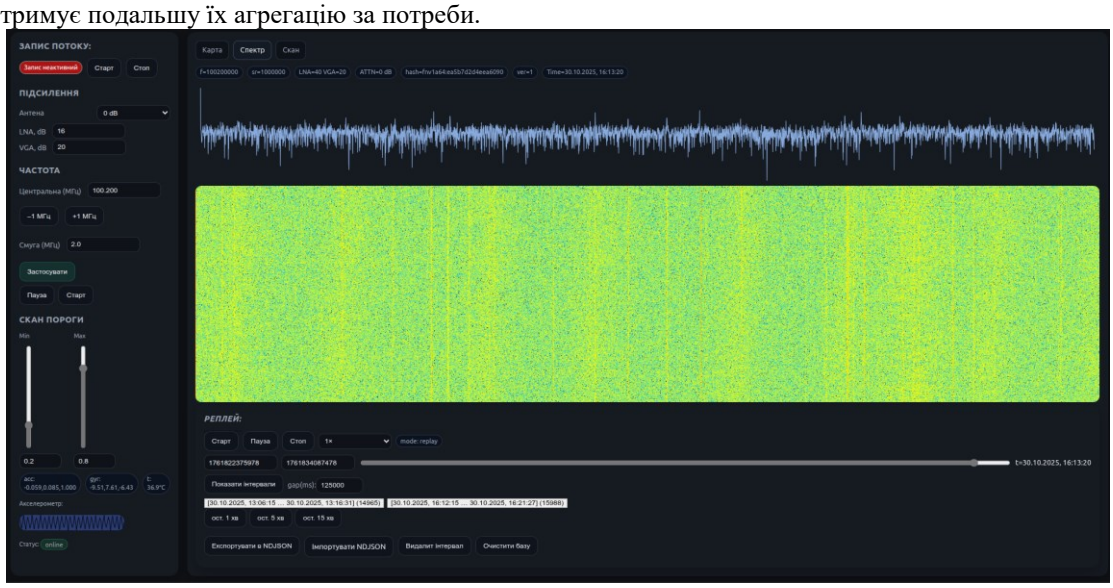

Рис. 3. Інтерфейс системи під відображення спектрограми

Спираючись на Рис.4. Можна виділити такі особливості користувацького інтерфейсу:

- **Панель налаштувань.** Дозволяє взаємодіяти із процесом що обробляє «сирі» дані та конфігурувати його одразу із користувацького простору. Так само клієнту надається можливість починати і зупиняти запис у базу даних, що у свою чергу надає можливість попередньо налаштувати систему і почати записувати отримані дані вже коли все підготовано. Застосування обраної конфігурації відбувається транзакційно: зміни вважаються внесеними тільки тоді, коли експортер відповів що успішно отримав команду та виконав її.

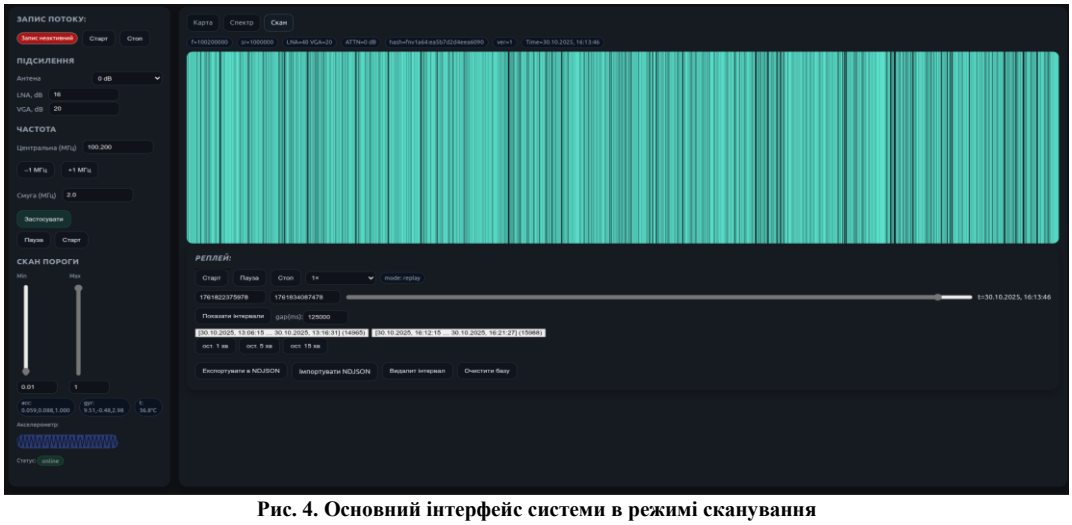

Рис. 4. Основний інтерфейс системи в режимі сканування

- **Live-режим.** Система дає стабільний потік спектральних кадрів зі сталою геометрією часу та частоти. У верхній частині інтерфейсу відображаються індикатори стану (*hash, ver, time*), посередині - спектр і водоспад. У разі локальних перевантажень потік обмежується у кількості повідомлень в секунду, залишаючи пріоритет за записом у базу даних.
- **Replay.** Для повторного аналізу використовується відтворення потоку на основі часових міток. Інтерфейс дозволяє завантажити інтервал, керувати швидкістю, переміщуватися на відрізку та зупиняти відтворення; графіки гарантовано очищуються при переході між режимами, що виключає змішування кадрів.
- **Експорт, імпорт та робота з даними.** Усі операції з даними - атомарні; експорт/імпорт обробляються потоком даних що переправляється в базу, тобто фактично копіювання файлів, без проміжних форматів. Така простота важлива задля уникнення спотворення даних при перекодуванні та архівації. Робота з проміжками базується на агрегації існуючих записів за певним руховим вікном розміру *gap* котре дозволяє групувати кадри на найближчі проміжки. Дані відрізки можна експортувати, видаляти. Так само наявний функціонал повної очистки бази, що зручно у випадку коли потрібно почати роботу “начисто”.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Запропонована архітектура - фокусується на автономності. Вона не потребує стійкого інтернету, працює при наявності живлення, антени і невеликого комп’ютера на кшталт Raspberry Pi. Послідовний потік даних джерело-база даних - користувач із єдиним форматом подій (1 подія на 1 рядок) надає незалежність від глобальних мереж і відтворюваність в обраних умовах. Дані фіксуються так, як вони приходять із експортера, всі кроки взаємодії зберігаються у тому ж потоці подій, що і спектри, що в свою чергу дозволяє отримувати ту саму конфігурацію системи, на якій записувався кожен збережений кадр.

Разом із тим система здатна до розширення та адаптації. Вона масштабується послідовно, без критичних змін: дані отримуються окремим процесом, котрий експортує їх в базу даних через посередника, на отриманих даних проводяться основні агрегації, а основні дані архівуються для збереження пам'яті на диску, подальший трафік природно переноситься у WebSocket з якого будується вся логіка інтерфейсу, проте виконується вона на стороні клієнта, що дозволяє уникнути основних навантажень. На основі цього підходу ми можемо як масштабувати цю систему у форматі розподілених систем, просто зв'язавши декілька таких приладів в єдину мережу використовуючи вже існуючий транспортний шар, або додати підтримку зовнішнього збереження та обробки даних при наявності глобальної мережі. Зміна формату обрахувань на рівні процесу що взаємодіє із периферією не ламає роботу решти шарів, необхідно лиш вказувати яку саме версію схеми передають далі.

Отже, особливість запропонованої системи - у поєднанні автономності з наявністю простору для розвитку: від простого офлайн-сканер, що працює в умовах відсутності зв'язку,до складних взаємопов'язаних систем, де кожен шар може бути модифікованим незалежно від інших. А перенесення більшості важких операцій на клієнта дозволяє зменшити вимоги до платформи на якій система розгортається. У подальшому можливо як рознесення системи на декілька платформ, так і додавання додаткового функціоналу (орієнтування у просторі, запис відеопотоку і т.д.).

Подяка

Автори висловлюють подяку за підтримку проекту НФДУ № 2023.04/0150 «Розробка комплексу для визначення положення та відносної потужності джерел радіовипромінювання та їх візуалізації».

Література

1. Frigo M., Johnson S. G. The Design and Implementation of FFTW3 // Proceedings of the IEEE. – 2005. – [Електронний ресурс]. – Режим доступу: https://www.fftw.org/fftw-paper-ieee.pdf. – Дата звернення: 06.11.2025.
2. Great Scott Gadgets. HackRF One Documentation / API Reference. – [Електронний ресурс]. – Режим доступу: https://greatscottgadgets.com/hackrf/ – Дата звернення: 06.11.2025.
3. Pasolini G., Fantini A., Rovati L., та ін. A Software Defined Radio Platform with Raspberry Pi and Applications // EUSIPCO. – 2016. – [Електронний ресурс]. – Режим доступу: https://www.eurasip.org/Proceedings/Eusipco/Eusipco2016/papers/1570255810.pdf. – Дата звернення: 06.11.2025.
4. Souryal M., Ranganathan M., Mink J., El Ouni N. Real-Time Centralized Spectrum Monitoring: Feasibility, Architecture, and Latency. – NIST, 2015. – [Електронний ресурс]. – Режим доступу: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=918631. – Дата звернення: 06.11.2025.
5. NDJSON - Newline-Delimited JSON. – [Електронний ресурс]. – Режим доступу: http://ndjson.org/. – Дата звернення: 06.11.2025.
6. Gaffney K. P., Owens A., Hipp D. R., та ін. SQLite: Past, Present, and Future // PVLDB. – 2022. – [Електронний ресурс]. – Режим доступу: https://www.vldb.org/pvldb/vol15/p3535-gaffney.pdf. – Дата звернення: 06.11.2025.

7. WHATWG. Server-Sent Events // HTML Living Standard. – [Електронний ресурс]. – Режим доступу: https://html.spec.whatwg.org/multipage/server-sent-events.html. – Дата звернення: 06.11.2025.
8. IETF. The WebSocket Protocol (RFC 6455). – 2011. – [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6455. – Дата звернення: 06.11.2025.
9. WHATWG. HTML Living Standard: Web Workers. – [Електронний ресурс]. – Режим доступу: https://html.spec.whatwg.org/multipage/workers.html – Дата звернення: 05.11.2025.
10. Khronos Group. WebGL 2.0 Specification. – [Електронний ресурс]. – Режим доступу: https://www.khronos.org/registry/webgl/specs/latest/2.0/ – Дата звернення: 06.11.2025.
11. W3C. Performance Timeline Level 2 (Recommendation). – [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/performance-timeline-2/ – Дата звернення: 05.11.2025.

References

1. Frigo M., Johnson S. G. The Design and Implementation of FFTW3 // Proceedings of the IEEE. – 2005. – [Digital resource]. – Access mode: https://www.fftw.org/fftw-paper-ieee.pdf. – Date of access: 06.11.2025.
2. Great Scott Gadgets. HackRF One Documentation / API Reference. – [Digital resource]. – Access mode: https://greatscottgadgets.com/hackrf/ – Date of access: 06.11.2025.
3. Pasolini G., Fantini A., Rovati L., et al. A Software Defined Radio Platform with Raspberry Pi and Applications // EUSIPCO. – 2016. – [Digital resource]. – Access mode: https://www.eurasip.org/Proceedings/Eusipco/Eusipco2016/papers/1570255810.pdf. – Date of access: 06.11.2025.
4. Souryal M., Ranganathan M., Mink J., El Ouni N. Real-Time Centralized Spectrum Monitoring: Feasibility, Architecture, and Latency. – NIST, 2015. – [Digital resource]. – Access mode: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=918631. – Date of access: 06.11.2025.
5. NDJSON - Newline-Delimited JSON. – [Digital resource]. – Access mode: http://ndjson.org/. – Date of access: 06.11.2025.
6. Gaffney K. P., Owens A., Hipp D. R., et al. SQLite: Past, Present, and Future // PVLDB. – 2022. – [Digital resource]. – Access mode: https://www.vldb.org/pvldb/vol15/p3535-gaffney.pdf. – Date of access: 06.11.2025.
7. WHATWG. Server-Sent Events // HTML Living Standard. – [Digital resource]. – Access mode: https://html.spec.whatwg.org/multipage/server-sent-events.html. – Date of access: 06.11.2025.
8. IETF. The WebSocket Protocol (RFC 6455). – 2011. – [Digital resource]. – Access mode: https://datatracker.ietf.org/doc/html/rfc6455. – Date of access: 06.11.2025.
9. WHATWG. HTML Living Standard: Web Workers. – [Digital resource]. – Access mode: https://html.spec.whatwg.org/multipage/workers.html – Date of access: 05.11.2025.
10. Khronos Group. WebGL 2.0 Specification. – [Digital resource]. – Access mode: https://www.khronos.org/registry/webgl/specs/latest/2.0/ – Date of access: 06.11.2025.
11. W3C. Performance Timeline Level 2 (Recommendation). – [Digital resource]. – Access mode: https://www.w3.org/TR/performance-timeline-2/ – Date of access: 05.11.2025.