

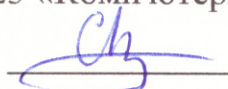
Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

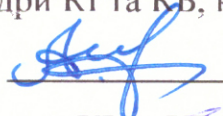
до випускної кваліфікаційної роботи бакалавра
на тему: «Проект комп'ютерної системи управління
комплексом радіоелектронної боротьби та розвідки»

Виконав: студент 4-го курсу групи КІ-21-2

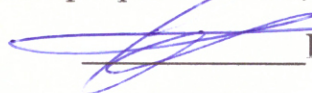
спеціальності 123 «Комп'ютерна інженерія»

 Владислав СКИДАНЧУК

Керівник: Доцент кафедри КІ та КБ, к.т.н.

 Олексій ШЕЛУХА

Рецензент: Доцент кафедри КІ та КБ, к.т.н., доцент

 Юрій БРОДСЬКИЙ

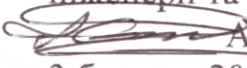
Житомир – 2025

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко
3 березня 2025 р.

ЗАВДАННЯ

на випускн кваліфікаційну роботу бакалавра

Студент: **Скиданчук Владислав Дмитрович**

Тема роботи: **«Проект комп'ютерної системи управління комплексом радіо-електронної боротьби та розвідки»**

затверджена Наказом університету від 3 березня 2025 р. № 79/с

Термін здачі студентом закінченої роботи 9 червня 2025 р.

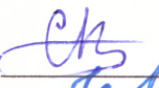
Вихідні дані роботи (зазначається предмет і об'єкт дослідження):
Комп'ютерна система управління комплексом РЕБ та РЕР, побудована на базі ESP32, Raspberry Pi 4 та протоколу MQTT.

Обладнання та програмне забезпечення:

1. Одноплатний комп'ютер Raspberry Pi 4 B
2. Мікроконтроллер ESP32 DevKit v1
3. Брокер повідомлень Mosquitto MQTT
4. Графічне середовище Node-RED, Flac
5. Середовище розробки Arduino IDE
6. Операційна система Raspberry Pi OS

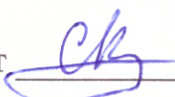
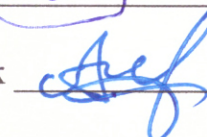
Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

Студент  Владислав СКИДАНЧУК
Керівник  Олексій ШЕЛУХА

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	03.03.2025	Виконано
2	Формулювання технічного завдання	10.03.2025	Виконано
3	Проектування структури	10.04.2025	Виконано
4	Опрацювання літератури	01.05.2025	Виконано
5	Створення проекту	13.05.2025	Виконано
6	Налаштування системи	20.05.2025	Виконано
7	Відлагодження	02.06.2025	Виконано
8	Написання пояснювальної записки	09.06.2025	Виконано
9	Захист	27.06.2025	

Студент  Владислав СКИДАНЧУК
Керівник  Олексій ШЕЛУХА

АНОТАЦІЯ

Анотація кваліфікаційної роботи бакалавра містить відомості про проєкт розробки комп'ютерної системи моніторингу та керування в радіочастотному середовищі. Робота включає пояснювальну записку обсягом 89 сторінки, що містить 25 рисунків, 2 таблиці, 11 найменувань у списку використаних джерел, розміщених на 2 сторінках.

Метою випускної кваліфікаційної роботи є розробка проєкту програмно-апаратного комплексу, здатного виявляти аномальні активності в заданому частотному діапазоні та ініціювати відповідь у вигляді сповіщення або активації виконавчих модулів.

У межах виконання проєкту визначено основні завдання для реалізації системи активного спектрального моніторингу; проаналізовано базові платформи й обґрунтовано вибір компонентів; розглянуто особливості застосування бездротової комунікації; описано реалізацію системи, що охоплює збір, обробку, збереження та передачу даних. Наведено результати тестування в умовах локального середовища, які підтверджують працездатність і ефективність запропонованої архітектури.

Ключові слова: МОНИТОРИНГ СПЕКТРА, РЕБ, СПЕКТРОАНАЛІЗАТОР, RASPBERRY PI, ESP32, MQTT, SQLITE, TELEGRAM-БОТ, АНАЛІЗ СИГНАЛІВ, АВТОМАТИЧНЕ КЕРУВАННЯ.

					ККІКБ.КР.Б-123-25-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проект комп'ютерної системи управ- ління комплексом радіоелектронної боротьби та розвідки	Літ.	Арк.	Аркушів
Розроб.		Скиданчук В.Д.		24.06				
Керівник		Шелуха О.О.		27.06			4	89
Реценз.		Бродський Ю.Б.		27.06		ФІКТ, Гр. КІ-21-2		
Н. контр.		Шелуха О.О.		28.06				
Затверд.		Єфіменко А.А.		22.08				

ANNOTATION

The abstract of the bachelor's qualification work provides information about a project aimed at developing a computer system for monitoring and controlling the radio frequency environment. The explanatory note consists of 89 pages, includes 25 figures, 2 tables, and 11 sources referenced across 2 pages.

The objective of this qualification project is to design a hardware-software complex capable of detecting abnormal activities within a defined frequency range and initiating a response, such as sending alerts or activating actuator modules.

Within the project scope, the main tasks for implementing an active spectrum monitoring system were outlined; fundamental platforms were analyzed, and the component selection was justified. The use of wireless communication was reviewed, and the system implementation was described, covering data acquisition, processing, storage, and transmission. The results of testing in a local environment confirmed the reliability and efficiency of the proposed architecture.

Key words: SPECTRUM MONITORING, ELECTRONIC WARFARE, SPECTRUM ANALYZER, RASPBERRY PI, ESP32, MQTT, SQLITE, TELEGRAM BOT, SIGNAL ANALYSIS, AUTOMATIC CONTROL.

.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП	9
-------------	---

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ СИСТЕМ РАДІОЧАСТОТНОГО МОНІТОРИНГУ ТА УПРАВЛІННЯ ЗАСОБАМИ ВПЛИВУ	11
---	-----------

1.1 Сучасні підходи до побудови систем радіочастотного моніторингу	11
1.1.1 Призначення та завдання систем РЧМ.....	11
1.1.2 Апаратні засоби для аналізу ефіру	11
1.1.3 Мікроконтролери та обчислювальні вузли в системах РЧМ.....	12
1.1.4 Методи виявлення спектральних аномалій	14
1.2 Програмно-апаратна основа побудови системи моніторингу та керування	15
1.2.1 Вибір мікроконтролерів для модулів збору та передачі даних	15
1.2.2 Програмна архітектура системи керування.....	16
1.2.3 Протоколювання, обробка критичних подій і валідація керуючих дій...	17
1.2.4 Резервування, стійкість і автономна робота в умовах втрати зв'язку	18
1.2.5 Інтерфейси керування та взаємодії з користувачем	19
1.2.6 Адаптація до завад, автономність та захист доступу	21
1.3 Формування архітектури комп'ютерної системи моніторингу та керування в динамічному радіочастотному середовищі	22
1.3.1 Логічна структура та функціональні блоки комп'ютерної системи	22
1.3.2 Логічна архітектура та зв'язки між компонентами системи	24
1.3.3 Адаптивність та масштабованість архітектури комп'ютерної системи ..	25
1.4 Постановка технічного завдання	26
1.4.1 Формулювання цілей і задач проєкту	26
1.4.2 Вимоги до реалізації	27
Висновок першого розділу	29

РОЗДІЛ 2 РОЗРОБКА І РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ТА КЕРУВАННЯ	30
--	-----------

2.1 Вибір середовища розробки та інструментів	30
2.2 Реалізація функціональних модулів	31
2.2.1 Реалізація збору даних із радіоспектра	31
2.2.2 Реалізація обробки даних на Raspberry Pi.....	32
2.2.3 Керування виконавчими модулями.....	34
2.2.4 Інтеграція з вебінтерфейсом та Telegram ботом	35
2.2.5 Забезпечення надійності, автономності та захисту дій	37
2.3 Реалізація інтерфейсів керування	38
2.3.1 Реалізація інтерфейсу керування через вебплатформу	38
2.3.2 Реалізація Telegram-бота для взаємодії з користувачем	40

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

2.3.3 Реалізація мобільного застосунку IoT MQTT Panel для iOS	42
2.4 Побудова системної логіки реагування	43
РОЗДІЛ 3 ВИБІР ОБЛАДНАННЯ, ПОБУДОВА, НАЛАГОДЖЕННЯ ТА	
ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ МОНІТОРИНГУ ТА	
КЕРУВАННЯ	47
3.1 Підготовка обчислювального вузла	47
3.1.1 Підготовка Python-оточення на Raspberry Pi.....	51
3.1.2 Налаштування MQTT-брокера Mosquitto	53
3.1.3 Конфігурація локальної бази даних SQLite.....	55
3.2 Підключення спектроаналізатора TinySA Ultra.....	58
3.3 Налаштування взаємодії між вузлами.....	60
3.3.1 Конфігурація MQTT-клієнта на Raspberry Pi	60
3.3.2 Налаштування MQTT-клієнта на ESP32.....	63
3.4 Telegram-бот: розгортання, інтеграція, сценарії використання.....	64
Висновок третього розділу	68
ВИСНОВКИ	70
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	72
ДОДАТКИ.....	74

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система

ПЗ – програмне забезпечення

РЧМ - радіочастотний моніторинг

SDR - software defined radio

МК – мікроконтролер

ЦОМ – центральний обчислювальний модуль

РЕБ – радіоелектронна боротьба

РЕР – радіоелектронна розвідка

JWT – JSON web token

КСМ – комп'ютерна система моніторингу

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Сьогодні цифрові технології активно змінюють як повсякденне життя, так і стратегії безпеки. Бездротовий зв'язок, автономні системи й платформи безпілотників стали звичним явищем, інтегрованим у цивільну інфраструктуру. Водночас їхнє використання у сфері оборони зростає. Через це відбувається розмиття між цивільним та військовим застосуванням техніки. У таких умовах потреба в надійних рішеннях, які здатні контролювати радіочастотне середовище, виявляти загрози та швидко реагувати на них зростає. Саме тому розробка системи, що поєднує аналіз ефіру та керування пристроями активного впливу на спектр радіочастот є важливим і своєчасним завданням.

Метою кваліфікаційної роботи є створення проєкту комп'ютеризованої системи управління комплексом радіоелектронної боротьби та розвідки. Система здатна здійснювати моніторинг радіоефіру, виявляти радіочастотні аномалії. До них належать раптові стрибки амплітуди сигналу, поява нехарактерних сигналів, нестабільність несучих частот, перекриття частот, а також короткі імпульсні збурення та зміни шумового фону (детальний опис подано в [додатку А](#)). Виявлені порушення обробляються з урахуванням впливу активних засобів, після чого система реагує керування відповідними jammer-модулями.

Реалізація проєкту передбачає проєктування архітектури системи, налаштування взаємодії між її модулями через MQTT протокол, розробку алгоритмів фільтрації інформації та створення інтерфейсу що здатна відображати стан системи й надавати керування в режимі реального часу.

Об'єктом дослідження є система, призначена для виявлення та реагування на зміни в спектрі радіочастоти. Вона охоплює загальні принципи обміну даними, дистанційного керування та аналізу сигналів у сфері зв'язку та технічного спостереження.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

У роботі основна увага зосереджена на засобах, які забезпечують стабільну взаємодію між складовими системи. Це, зокрема, мікроконтролери, протоколи передачі даних, алгоритми обробки сигналів та інтерфейс, який дозволяє користувачу спостерігати за станом системи та керувати її роботою.

У процесі виконання кваліфікаційної роботи було розроблено прототип, який поєднує модуль аналізу радіочастотного спектра, центральний вузол на основі Raspberry Pi та програмну частину для передачі команд виконавчим елементам. Система розрахована на автономне використання і може не потребує додаткових змін для практичного застосування або слугувати основою для подальших модифікацій та інтеграції в більші системи.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ СИСТЕМ РАДІОЧАСТОТНОГО МОНІТОРИНГУ ТА УПРАВЛІННЯ ЗАСОБАМИ ВПЛИВУ

1.1 Сучасні підходи до побудови систем радіочастотного моніторингу

Системи радіочастотного моніторингу стали ключовими елементами сучасної інфраструктури контролю за ефіром. Вони дозволяють відстежувати спектральне середовище, виявляти нові сигнали та фіксувати їхні аномалії. Вони дозволяють відстежувати спектральне середовище, виявляти нові сигнали та фіксувати їхні аномалії. Подібні рішення використовуються у телекомунікаціях, оборонних технологіях, цивільній безпеці та системах радіомоніторингу в умовах перевантаженого або зашумленого середовища.

1.1.1 Призначення та завдання систем РЧМ

У різних галузях - від цифрових телекомунікацій до систем технічного захисту – РЧМ виконують роль первинного джерела спектральної інформації. Їх призначення полягає у зборі, фіксації та подальшій інтерпретації даних про активність у визначених частотних діапазонах. Залежно від контексту застосування, ці системи можуть бути орієнтовані як на широкосмугове сканування, так і на моніторинг специфічних протоколів або вузьких каналів зв'язку.

Вони забезпечують не лише пасивне спостереження, а й активну взаємодію з іншими модулями - наприклад, аналітичними платформами або виконавчими пристроями. Такі комплекси особливо ефективні у динамічному або агресивному спектральному середовищі, де характер сигналів постійно змінюється. У цих умовах важливою є здатність системи адаптувати критерії виявлення і забезпечити безперервне постачання даних на вищий рівень керування.

1.1.2 Апаратні засоби для аналізу ефіру

Ядром будь-якої системи радіочастотного моніторингу є апаратні компоненти, що безпосередньо приймають та перетворюють сигнали. Від вибору таких засобів залежить як спектральне охоплення, так і точність виявлення та швидкість

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

обробки. Серед основних характеристик – частотний діапазон, роздільна здатність по частоті, швидкість сканування та інтерфейс підключення.

Для побудови недорогих, але функціональних РЧМ-рішень часто використовуються модулі RX5808 - аналогові приймачі, що працюють в діапазоні 5,8 ГГц. Вони здатні змінювати канали через SPI-інтерфейс і надають аналоговий рівень сигналу (RSSI) для оцінки потужності. Ці модулі добре зарекомендували себе у проєктах з невисокими вимогами до точності або в ситуаціях, де важлива автономність.

Для ширших завдань застосовуються SDR-приймачі, зокрема RTL-SDR, HackRF One, LimeSDR. Вони забезпечують набагато гнучкіший аналіз: від прийому в діапазоні від кількох сотень кГц до кількох ГГц, до виведення IQ-даних із високою точністю. Такі рішення потребують більшої обчислювальної потужності. Водночас вони дають змогу використовувати складніші алгоритми аналізу, зокрема спектральне згортання, кластеризацію та фільтрацію шуму.

В окремих випадках застосовуються інтегровані мікросхеми з вбудованими аналогово-цифровими перетворювачами, які надають RSSI або спектральні дані напряму у цифровому форматі. Наприклад, RFM69, CC1101, або більш складні SoC на базі ESP32 з функцією Wi-Fi-сканування.

Незалежно від типу, обраний модуль повинен мати достатню чутливість, стабільність частотного генератора, та інтерфейс для інтеграції з логікою обробки - найчастіше SPI або I2C.

1.1.3 Мікроконтролери та обчислювальні вузли в системах РЧМ

Зібрані з сенсорів дані потребують обробки та інтерпретації – цю функцію забезпечують мікроконтролери та обчислювальні вузли. Вони виконують роль керуючих елементів системи. Також відповідають за конфігурацію приймачів, відбір потрібних даних та виведення MQTT-повідомлень.

У якості мікроконтролера часто застосовується ESP32, який має достатню продуктивність, апаратну підтримку SPI, I2C та вбудовану підтримку Wi-Fi. Цей чип дозволяє паралельно обробляти кілька потоків даних за допомогою FreeRTOS

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

та передавати результати до брокера MQTT. Завдяки енергоефективності та компактності ESP32 оптимально підходить для автономних розвідок спектра.

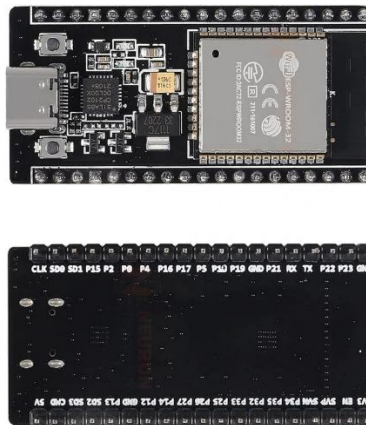


Рисунок 1.1 - Мікроконтролер ESP32

У більш відповідальних задачах доцільно застосовувати STM32. Ці мікроконтролери забезпечують високу надійність, стабільну роботу з точними таймерами та широкими можливостями налаштування периферії. Вони дозволяють реалізувати складні логічні операції, керування частотами та обробку сигналів із мінімальними затримками. Також доступна апаратна підтримка шифрування та прямого доступу до пам'яті (DMA), що підвищує ефективність системи.

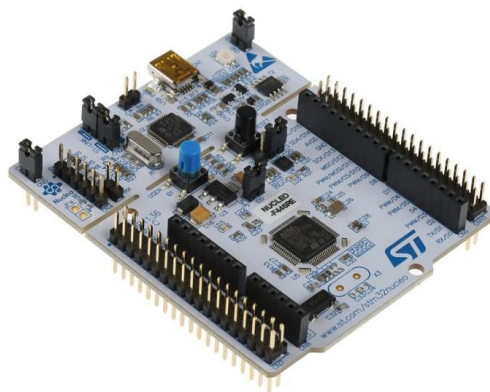


Рисунок 1.2 - Мікроконтролер на базі STM32

У центрі системи зазвичай розміщується Raspberry Pi - одноплатний комп'ютер з повноцінною ОС Linux. Його задача - отримати попередньо оброблені дані, виконати аналіз спектра, виявити аномалії, візуалізувати інформацію та при потребі - надіслати команду на вмикання виконавчих пристроїв. Через графічний

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерфейс користувач отримує доступ до повної картини роботи комплексу в реальному часі.



Рисунок 1.3 - Одноплатний комп'ютер Raspberry Pi 4 Model B

Ключову роль відіграє формат взаємодії між вузлами - найчастіше використовують протокол MQTT, який забезпечує легкий та швидкий обмін повідомленнями між пристроями в межах локальної або глобальної мережі.

1.1.4 Методи виявлення спектральних аномалій

У сучасних системах радіомоніторингу важливо не лише приймати та обробляти сигнали, а й виявляти відхилення, що можуть свідчити про потенційні загрози. Спектральні аномалії — це зміни у частоті, амплітуді або часовій структурі сигналу, що не характерні фону. Вони виступають ключовими індикаторами порушень у радіоефірі. Найпростіші методи виявлення базуються на порогових значеннях амплітуди: якщо сигнал перевищує поріг, фіксується аномалія. Часто використовують ковзне середнє або згладжування для уточнення змін фону.

У складніших реалізаціях застосовують аналіз частотних або часових вікон, похідні по частоті для фіксації нестабільної несучої та зсувів, притаманних рухомих джерелам. Для імпульсних збурень використовують буфери історії та високу частоту дискретизації.

Зміни спектральної щільності шуму — ще один індикатор порушення. Нехарактерні сигнали, що не відповідають шаблонам, виявляються через класифікацію або кореляційний аналіз. За наявності ресурсів — через алгоритми машинного навчання.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Обрані методи мають відповідати апаратним можливостям системи. Баланс між точністю, швидкістю обробки та ресурсами є критично важливим для ефектвної реалізації.

1.2 Програмно-апаратна основа побудови системи моніторингу та керування

У цьому підрозділі розглядаються основні елементи, на яких базується робота системи. Йдеться про апаратні компоненти, мікроконтролери та пристрої взаємодії. Також аналізуються програмні засоби, які забезпечують обмін даними і керування. Система має працювати стабільно, бути адаптивною та здатною до автономного функціонування. Для цього важливо підібрати правильну архітектуру, забезпечити сумісність між модулями й обрати надійні протоколи передачі інформації.

1.2.1 Вибір мікроконтролерів для модулів збору та передачі даних

Фізична частина системи побудована на поєднанні енергоефективних мікроконтролерів і спеціалізованих модулів. Основу становить центральний вузол - одноплатний комп'ютер Raspberry Pi 4, що виконує обчислення, обробку сигналів і візуалізацію даних. Цей пристрій обрано через його компактність, доступність та підтримку широкого спектра інтерфейсів і бібліотек.

Для попереднього збору та обробки даних використовуються мікроконтролери типу ESP32. Вони забезпечують початковий аналіз сигналів, взаємодію з сенсорними модулями та передачу даних до центрального вузла. У системі можуть застосовуватись кілька таких мікроконтролерів залежно від топології комплексу.

Вибір мікроконтролерів зумовлений необхідністю гнучкого управління й швидкого реагування. Вони здатні працювати в автономному режимі, мають низьке енергоспоживання та розвинену підтримку протоколів, зокрема MQTT. Це дозволяє реалізувати ефективну взаємодію між окремими сегментами системи.

До апаратної бази також входять модулі живлення, перетворювачі напруги, комунікаційні інтерфейси та елементи захисту. Таке компонування забезпечує стабільну роботу навіть у складних умовах.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

1.2.2 Програмна архітектура системи керування

Програмна частина системи виконує функцію зв'язуючої ланки між обчислювальним вузлом, сенсорними модулями та виконавчими пристроями. Центральний обчислювальний модуль базується на платформі Raspberry Pi. Для системного керування обрано середовище на основі Linux, що забезпечує стабільну роботу, доступ до системних ресурсів, підтримку бібліотек для обробки даних і просту інтеграцію з мережевими сервісами. Основні компоненти програмної архітектури поділяються на три рівні: передача даних, обробка інформації та інтерфейс взаємодії.

- Передача даних:

Комунікація між ESP32 і ЦОМ здійснюється за допомогою MQTT. Цей протокол обміну повідомленнями забезпечує асинхронну публікацію та підписку на теми, що дозволяє будувати подієво-орієнтовану архітектуру. Кожен вузол працює як окремий агент, надсилаючи повідомлення про сканування ефіру або приймаючи інструкції на вмикання чи вимикання засобів РЕБ. MQTT-брокер, встановлений на ЦОМ, виступає посередником у передачі цих даних і слугує точкою синхронізації між усіма компонентами системи.

- Обробка інформації:

На рівні обробки даних реалізовано алгоритми фільтрації, які дозволяють виключити сигнали, що потрапляють у заздалегідь визначені діапазони роботи РЕБ. Це дає змогу уникнути помилкового аналізу або повторного виявлення активних частот, які вже знаходяться під контролем. Програма також ідентифікує типи аномалій і зіставляє їх з історією сигналів. Сценарії реакції формуються залежно від типу виявленої події та прив'язуються до відповідних MQTT-команд.

- Користувацький інтерфейс:

Для керування системою створено два паралельні канали доступу:

- Вебінтерфейс на NODE – RED забезпечує графічне відображення результатів сканування, дозволяє вручну перемикає частоти, перегля-

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

дати лог подій, вводити діапазони фільтрації та отримувати статус роботи пристроїв у реальному часі. Всі дії відображаються інтерактивно через браузер, що зручно для локального або віддаленого доступу.

- Telegram-бот, розгорнутий на тому ж ЦОМ, дозволяє користувачеві надсилати текстові команди з мобільного пристрою. Реалізовано набір інструкцій для запиту стану системи, запуску повторного аналізу, отримання повідомлення про останню виявлену аномалію, а також активації або деактивації відповідних частот. Авторизація здійснюється за Telegram ID.

Такий розподіл дозволяє користувачам працювати із системою в різних контекстах: зручніше з комп'ютера — через браузер, мобільно — через месенджер. Це підвищує гнучкість експлуатації і спрощує адаптацію в польових умовах або в рамках стаціонарного розгортання.

1.2.3 Протоколювання, обробка критичних подій і валідація керуючих дій

Журналювання подій спрощує контроль роботи системи та аналіз збоїв. Наприклад, при втраті зв'язку з центральним вузлом локальний модуль автономно записує дані в тимчасову пам'ять, зберігаючи мітки часу, джерело ініціації та опис дії. Це дозволяє інженеру відтворити послідовність подій і визначити причину проблеми.

У нестабільному радіочастотному середовищі просте логування недостатнє. Алгоритми фільтрації, ймовірно, ефективно виявляють критичні ситуації, такі як стрибки амплітуди сигналу чи повторні команди за короткий час. У таких випадках дія блокується, користувач отримує сповіщення, а журнал фіксує інцидент. Це забезпечує швидке реагування та збереження даних для подальшого аналізу.

Чи здатна система захиститись від помилок? Кожна команда проходить кілька перевірок: цілісність даних, відповідність параметрів (наприклад, частота в дозволеному діапазоні) та узгодженість із поточним режимом. Наприклад, запит на

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

зміну частоти під час роботи РЕБ-модуля може бути проігнорований або відправлений на підтвердження до ЦОМ. Такий багаторівневий підхід мінімізує ризик некоректних дій. Для запобігання перевантаженням програмне забезпечення обмежує частоту команд. Повторні запити з високою інтенсивністю позначаються як потенційно небезпечні, що знижує ймовірність збоїв чи зловмисного втручання.

Стійкість до збоїв забезпечується резервними сценаріями. При втраті зв'язку з ЦОМ модулі переходять в автономний режим, зберігаючи налаштування до відновлення комунікації. На практиці це дозволяє, наприклад, уникнути втрати контролю протягом кількох хвилин.

Ще один рівень безпеки — перевірка контексту. Наприклад, якщо команда змінити частоту надходить під час роботи РЕБ-модуля, алгоритми, ймовірно, проігнорують її або перенаправлять до ЦОМ для підтвердження. Такий підхід, на практиці, забезпечує багаторівневу валідацію, коли критичні дії потребують схвалення вищого рівня керування чи оператора. Інтерфейс системи, доступний через вебплатформу або Telegram-бота, відображає поточний стан і журнал подій. Це допомагає швидко виявляти аномалії, як-от некоректні команди чи збої.

Завдяки взаємодії журналювання, валідації та адаптивної реакції, як показують тести, система залишається надійною навіть у складних умовах, забезпечуючи прозорість і можливість масштабування без значних змін архітектури.

1.2.4 Резервування, стійкість і автономна робота в умовах втрати зв'язку

У розподілених системах моніторингу й керування важливою вимогою є здатність підтримувати стабільну роботу навіть за умов перебоїв зв'язку. Особливо це стосується середовищ із високим рівнем радіочастотних завад, де може виникати втрата з'єднання між окремими вузлами й центральною керуючою ланкою.

У цій розробці кожен пристрій збору та аналізу даних, побудований на базі ESP32, здатен працювати в автономному режимі. Якщо ЦОМ на основі Raspberry Pi 4 тимчасово недоступний, локальні вузли продовжують фіксувати спектральну активність, зберігаючи дані в енергонезалежній пам'яті. Після відновлення зв'язку

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

накопичена інформація передається на ЦОМ із часовими мітками, що забезпечує точне відтворення хронології подій у радіоефірі. Окремо налаштована логіка автономного реагування. Наприклад, при виявленні сильного джерела перешкод, модуль може самостійно перейти на інший частотний діапазон або активувати локальні алгоритми зниження енергоспоживання. У критичних випадках пристрій переходить у режим обмеженої активності. Це призводить до зниження частоти сканування, мінімізації обсягу записів а також деактивує допоміжні інтерфейси.

Надійність системи посилюється механізмом блокування управлінських команд за відсутності підтвердження від ЦОМ. Наприклад, команда на активацію РЕБ у момент втрати зв'язку зберігається в буфері та виконується лише після верифікації. Такий підхід знижує ризик помилкових чи несанкціонованих дій, що є критично важливим для радіоелектронної розвідки.

Окрему увагу приділено резервуванню конфігурацій. Кожен вузол має заздалегідь налаштовані сценарії для обробки типових аномалій, таких як короткочасна втрата зв'язку чи поява непередбачених сигналів у спектрі. Наприклад, зафіксувавши перевищення частоти вхідних команд, пристрій автоматично призупиняє їх обробку, що запобігає перевантаженню системи.

З технічного погляду, ESP32 дозволяє реалізувати як базове логування, так і автономну обробку подій завдяки наявності енергонезалежної пам'яті та підтримці периферійних інтерфейсів. Raspberry Pi 4 виконує функції агрегації, аналізу та інтеграції даних, а також контролює критичні дії в системі. Отже, поєднання резервування, локальної обробки та адаптивних механізмів забезпечує стабільну роботу системи в умовах нестабільного радіочастотного середовища.

1.2.5 Інтерфейси керування та взаємодії з користувачем

Зручність і надійність взаємодії користувача з технічною системою визначають не лише ефективність керування, а й оперативність реагування на позаштатні

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

ситуації. В межах цього проєкту буде реалізовано одразу кілька незалежних інтерфейсів: вебінтерфейс, Telegram-бот та мобільний застосунок на базі IoT MQTT Panel.

Вебінтерфейс, побудований на Node-RED, є центральним елементом виводу даних у режимі реального часу. Він забезпечує графічне відображення спектру, статус модулів, журнал подій і дозволяє керувати засобами впливу, зокрема ввімкненням і вимкненням окремих частот. Завдяки розширенням на базі JavaScript можлива адаптація до сценаріїв конкретного користувача: наприклад, активація логічних фільтрів, зміна чутливості тригерів або налаштування реакції на повторювані аномалії. Інтерфейс також виводить контекстну інформацію про джерело події, рівень сигналу, його стабільність та приблизну відстань до передавача.

Telegram-бот слугує допоміжним інструментом для інформування оператора та дистанційного керування системою. Він, ймовірно, інтегрується з MQTT-брокером через Flask-сервер, який діє як посередник між хмарним месенджером і локальною інфраструктурою. Бот сповіщає про критичні зміни, наприклад, відхилення в шумовому фоні чи втрату зв'язку з модулем. Оператор може підтвердити або відхилити важливі команди, скажімо, перемикання режиму роботи РЕБ-модуля, не відкриваючи вебінтерфейс. За слабого інтернетного зв'язку перевага надається коротким повідомленням із кнопками для швидкої відповіді, що зменшує затримки. Доповнює цей канал мобільний додаток IoT MQTT Panel. Через MQTT-з'єднання він відображає актуальні параметри системи й дає змогу надсилати команди без складної авторизації. Такий підхід зручний у польових умовах, коли потрібно швидко оцінити ситуацію чи активувати сценарій, наприклад, зупинити сканування або перейти на резервну частоту.

Сумісність між усіма інтерфейсами досягається завдяки уніфікованій структурі MQTT-повідомлень та ідентифікації дій за типами подій і джерелами. Це дозволяє забезпечити стабільний і взаємодоповнюючий канал керування без дублю-

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

вання функцій. Наприклад, команди з Telegram-бота можуть бути автоматично відображені в Node-RED, а повідомлення про збої з додатку — переспрямовані у Telegram для термінової реакції.

Таким чином, багатоканальна модель інтерфейсів не лише підвищує зручність використання, але й сприяє стійкості системи: у разі виходу з ладу одного з каналів інші залишаються активними, забезпечуючи резервну функціональність.

Основна реалізація передбачає розробку Telegram-бота та мобільний MQTT-клієнт. Архітектура системи передбачає можливість розширення функціоналу шляхом інтеграції web-інтерфейсу для аналітики та візуалізації даних, що розглядається як перспективний напрямок розвитку.

1.2.6 Адаптація до завад, автономність та захист доступу

Радіочастотне середовище часто буває нестабільним. Імовірність перешкод, збоїв чи втрат зв'язку з центральним вузлом потребують гнучких сценаріїв реагування. Кожен модуль системи може самостійно оцінювати локальну ситуацію, зберігати параметри роботи та активувувати алгоритми, якщо з'єднання з MQTT-брокером втрачено. Наприклад, при перевантаженні певного діапазону, локальний модуль автоматично знижує активність або переходить у режим очікування, доки канал не стабілізується.

Залежно від типу події, платформа може змінювати частоти, ізолювати окремі вузли чи вмикати засоби придушення. Реакція ґрунтується не лише на фіксованих правилах, а й на контексті. Система перевіряє, чи команда була очікуваною, чи сигнал надійшов із дозволеної IP-адреси, і чи не минув час виконання попередньої дії. Такий підхід, ймовірно, забезпечує працездатність у складних умовах.

Особливу увагу приділено захисту доступу. Оскільки система допускає дистанційне керування, важливо запобігти втручанню третіх осіб. Для цього реалізовано кілька рівнів безпеки.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

По-перше, кожен канал комунікації потребує автентифікації - Telegram-бот працює лише з авторизованими ID, інтерфейс Node-RED має обмеження за IP та JWT-токенами.

По-друге, критичні команди, такі як перемикання режимів РЕБ або скидання параметрів, додатково перевіряються на рівні ЦОМ. Команда не проходить, якщо її параметри не відповідають допустимим або вона не була підтверджена оператором.

Система веде журнал успішних та відхилених запитів. Це дає змогу виявити підозрілу активність, наприклад, логічна DDoS-атака, метою якої є дестабілізація вузлів керування або спотворення реального стану системи.

Поєднання автономної логіки, перевірок достовірності, захищеного зв'язку та обмеження прав доступу створює надійний захист. Навіть у разі втрати зв'язку чи часткової компрометації вузлів, система залишається працездатною. Це особливо важливо в польових умовах.

1.3 Формування архітектури комп'ютерної системи моніторингу та керування в динамічному радіочастотному середовищі

Даний розділ присвячено побудові архітектури комп'ютерної системи, яка забезпечує моніторинг та керування у змінному радіочастотному середовищі. Увага зосереджена на взаємодії ключових апаратних і програмних компонентів, включно з пристроями збору спектральних даних, обчислювальними вузлами, інтерфейсами керування та каналами передачі повідомлень.

Розроблена структура системи дозволяє забезпечити злагоджену взаємодію між модулями, адаптацію до змін в ефірі, а також автоматичне або операторське втручання у разі виявлення радіочастотних аномалій. У межах розділу буде проведено логічне структурування компонентів, описано принципи побудови взаємозв'язків та запропоновано обґрунтовану топологію реалізації.

1.3.1 Логічна структура та функціональні блоки комп'ютерної системи

Для реалізації системи моніторингу та керування в умовах змінного радіочастотного середовища необхідно забезпечити чітке розмежування обов'язків між

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

окремими функціональними блоками. Така структуризація дозволяє гнучко масштабувати систему, забезпечити надійність, а також адаптувати її під конкретні сценарії використання - від автономного моніторингу до інтеграції у більші інформаційно-аналітичні комплекси.

На логічному рівні система поділяється на п'ять основних блоків:

1. Блок збору та попередньої обробки даних

- Складається з пристроїв первинного моніторингу спектра: спектроаналізатора TinySA Ultra або інших модулів, здатних виявляти сигнали в заданому частотному діапазоні. Блок формує сирі або агреговані дані зі спектра ефіру, додає часові мітки, та передає результати на вищий рівень

2. Обчислювальний вузол.

- В ролі центрального обчислювального модуля виступає Raspberry Pi 4. Він приймає дані з блоків збору, виконує спектральний аналіз, виявлення аномалій та сценарне реагування. Саме тут реалізовані алгоритми фільтрації, протоколювання, контроль допустимих частот, валідація керуючих дій та взаємодія з базою даних. Обчислювальний вузол виступає як головний координатор системи.

3. Інтерфейсний блок.

- Забезпечує доступ користувача до керування системою. Сюди входить вебінтерфейс, Telegram-бот, а також IoT-застосунок для мобільного пристрою. Кожен інтерфейс має власні функції - від моніторингу подій до активації частот, запуску режимів та отримання журналів.

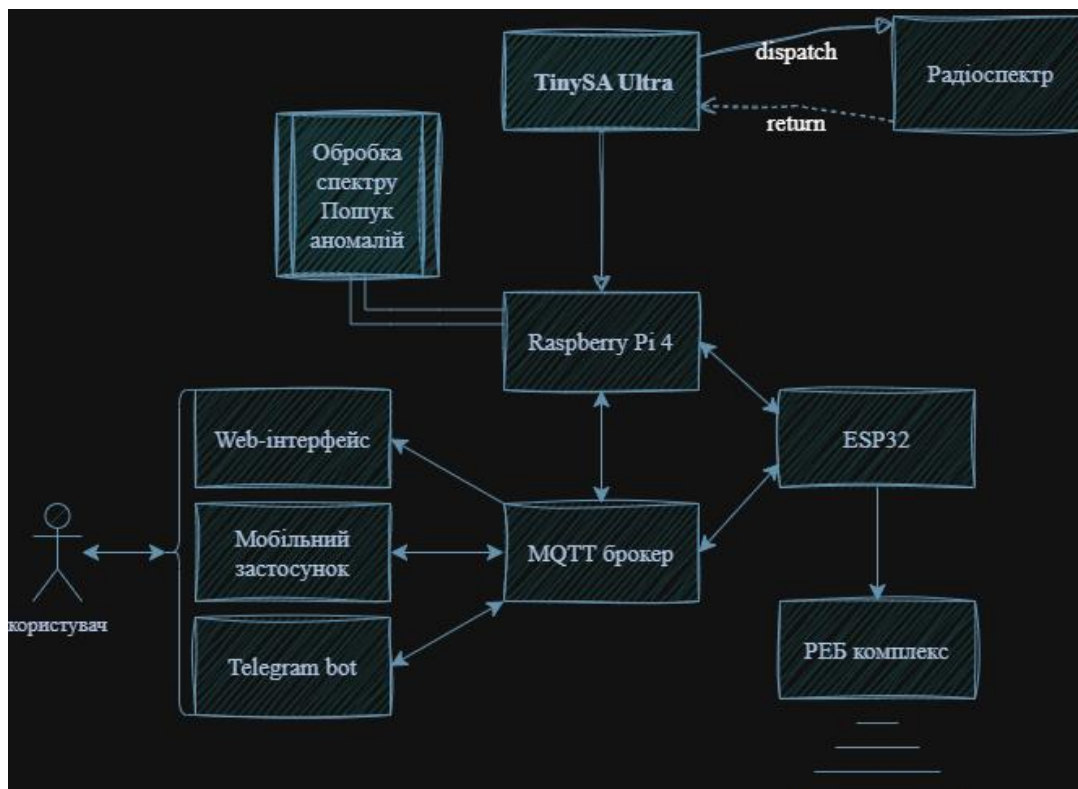
4. Блок керування засобами впливу.

- Призначений для активації виконавчих пристроїв (наприклад, jamming-модулів), які можуть подавати вплив у певному частотному діапазоні. Контроль здійснюється на основі подій, виявлених обчислювальним вузлом, з урахуванням політик, налаштувань користувача та допустимих сценаріїв.

5. Блок зв'язку.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

- Відповідає за комунікацію між компонентами. Може базуватись на MQTT, HTTP-запитах або локальному з'єднанні через UART/I2C. Забезпечує доставку команд і зворотного зв'язку, а також буферизацію у разі втрати з'єднання.



- Рисунок 1.4 - Логічна структура комп'ютерної системи керування

1.3.2 Логічна архітектура та зв'язки між компонентами системи

Логічна архітектура розробленої комп'ютерної системи передбачає чітко структуровану взаємодію між усіма її складовими. Кожен компонент виконує окрему роль у процесі моніторингу радіочастотного середовища, аналізу спектра, і формування реакційної відповіді. Система побудована за принципом модульної взаємодії, де центральну координацію здійснює вузол на базі Raspberry Pi 4.

Для прийому та попередньої обробки сигналів використовується спектральний аналізатор TinySA Ultra. Його основне завдання - фіксація стану радіоефіру та передача відповідних даних у цифровому форматі на центральний вузол. Подальший аналіз виконується програмними модулями на самому Raspberry Pi 4. Вони

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

здійснюють пошук аномалій, таких як нетипові частотні пульсації, перекриття каналів або імпульсні перешкоди.

Після виявлення потенційної загрози або змін у спектрі, система ініціює обмін інформацією з інтерфейсними модулями. Комунікація між підсистемами відбувається через MQTT-брокер, який забезпечує передачу повідомлень між Raspberry Pi, виконавчими модулями на базі ESP32, а також інтерфейсами керування.

ESP32 отримує керуючі команди та, у разі необхідності, активує або змінює параметри роботи РЕБ-комплексу. Наприклад, це може бути зміна частоти глушіння або увімкнення певного режиму протидії. Водночас система веде облік усіх подій через журналювання та забезпечує механізми перевірки коректності отриманих команд.

Користувач взаємодіє із системою за допомогою одного з трьох доступних інтерфейсів: веб-додатку, Telegram-бота або мобільного застосунку на базі IoT MQTT Panel. Усі вони підключаються до брокера, звідки отримують поточні дані про стан системи, включаючи виявлені загрози, лог подій та статус виконавчих модулів. Користувач також може надсилати команди керування, які проходять перевірку та валідацію на рівні ЦОМ перед їх передачею на нижчий рівень.

На [рисунку 1.4](#) представлено логічну схему комп'ютерної системи, яка відображає потоки даних та напрямки керування між компонентами. Така структура дає змогу масштабувати систему, додавати нові модулі, адаптувати її до нових умов без критичних змін у загальній архітектурі.

1.3.3 Адаптивність та масштабованість архітектури камп'ютерної системи

Описана система орієнтована на роботу в різних непостійних умовах. У зв'язку з цим, під час проєктування архітектури особливу увагу приділено її адаптивності та масштабованості. Це дає змогу використовувати систему в різних сценаріях без необхідності повної перебудови її структури. Адаптивність забезпечується модульною організацією. Кожен блок: спектроаналізатор, обчислювальний

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

модуль, інтерфейс або виконавчий пристрій - працює автономно, проте злагоджено взаємодіє з іншими. Наприклад, замість спектроаналізатора TinySA Ultra можливо використовувати інші пристрої, якщо вони передають дані у підтримуваному форматі. У такому разі не потребується зміна центрального модуля чи загальної логіки обробки.

За схожим принципом можна замінити будь-який виконавчий сегмент системи. На заміну мікроконтролеру ESP32 можна застосувати інший пристрій, сумісний із протоколом MQTT або HTTP, до прикладу контролери на базі процесора STM. Такий підхід підвищує гнучкість і дає змогу адаптувати систему під специфічні вимоги різних об'єктів.

Масштабованість системи проявляється у можливості підключення додаткових джерел спектральних даних. Наприклад, декілька аналізаторів можуть працювати паралельно та передавати дані на центральний вузол. Raspberry Pi 4 приймає ці дані, агрегує їх, виявляє аномалії та формує відповідь. У такий спосіб можливо охопити більшу зону моніторингу або підвищити точність виявлення сигналів.

Оновлення конфігурації можливе без перезавантаження пристроїв. Центральний вузол може зчитувати нові параметри з конфігураційного файлу в реальному часі. Це зручно для впровадження нових сценаріїв реагування чи налаштування параметрів без фізичного доступу до обладнання.

1.4 Постановка технічного завдання

1.4.1 Формулювання цілей і задач проєкту

Проектування комп'ютерної системи моніторингу та керування в радіочастотному середовищі потребує чіткого визначення цілей і задач. Це формує основу для побудови архітектури, підбору компонентів та реалізації ключових функцій.

Метою є створення діючого прототипу системи, здатної проводити моніторинг ефіру, виявляти спектральні аномалії та реагувати на загрози. Система повинна працювати автономно, бути гнучкою до змін конфігурації та інтегровуватись у ширші інформаційно-аналітичні комплекси.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Для досягнення цієї мети визначено такі задачі:

- Обґрунтувати вибір апаратних та програмних компонентів. Зокрема - аналізаторів спектра, вузлів обробки даних, засобів керування та взаємодії з користувачем.
- Побудувати логічну архітектуру системи. Вона має забезпечити чітку взаємодію між блоками збору, аналізу, реагування та інтерфейсами.
- Реалізувати механізми виявлення аномалій: стрибків амплітуди, перекриття частот, імпульсних збурень.
- Запровадити протоколювання подій, обробку критичних ситуацій, фільтрацію сигналів та перевірку команд.
- Створити кілька каналів керування - Telegram-бот та мобільний застосунок IoT MQTT Panel.
- Реалізувати захист від помилкових або шкідливих дій. Це включає перевірку команд, обмеження частоти запитів та резервування на випадок втрати зв'язку.
- Забезпечити масштабованість системи. Її компоненти мають легко доповнюватися без потреби у зміні основної архітектури.

Описані задачі сформульовано з урахуванням реальних умов роботи системи. Вони охоплюють усі рівні — від збору сигналів до передачі команд виконавчим пристроям.

1.4.2 Вимоги до реалізації

Для виконання проєкту слід чітко визначити технічні, програмні та функціональні вимоги. Вони забезпечують цілісність архітектури, стабільність роботи та можливість адаптації в різних умовах.

Апаратні вимоги:

- Спектроаналізатор: пристрій TinySA Ultra, здатний працювати в широкому діапазоні частот, з цифровим інтерфейсом передачі даних.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

- Центральний обчислювальний модуль: Raspberry Pi 4, який забезпечує обробку спектральних даних, керування процесами та взаємодію між пристроями.
- Виконавчі елементи: мікроконтролери ESP32, що приймають керуючі команди та активують модулі впливу (наприклад, ретрансляцію, глушіння).
- Комунікаційні інтерфейси: з'єднання на основі MQTT, UART, I²C або локального Ethernet/Wi-Fi-зв'язку.

Програмні вимоги:

- Операційна система на ЦОМ: Raspberry Pi OS із встановленими модулями Python, брокером Mosquitto, вебсервером Flask та інтерфейсними модулями.
- Telegram API для віддаленої взаємодії через бот.
- Мобільний застосунок: клієнт MQTT (наприклад, IoT MQTT Panel) для iOS/Android з попередньо визначеною структурою тем.

Функціональні вимоги:

- Можливість прийому даних від спектроаналізатора з часовими мітками.
- Виявлення визначених аномалій на основі алгоритмів аналізу.
- Протоколювання всіх дій та збереження журналів у локальну базу даних.
- Активація виконавчих пристроїв відповідно до сценаріїв реагування.
- Дистанційне керування з рівнями доступу для користувачів.
- Стійкість до втрати зв'язку, з переходом у автономний режим роботи.
- Гнучка адаптація системи до різних джерел вхідних сигналів та каналів передачі команд.

Також врахувати обмеження системи, частотний діапазон безпосередньо залежить від технічних характеристик використовуваного спектроаналізатора, обчислювальні ресурси центрального модуля обмежують можливості одночасної обробки великого обсягу даних, а затримки в передаванні команд можуть виникати в умовах нестабільного мережевого з'єднання або в разі втрати MQTT-зв'язку.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Висновок першого розділу

Перший розділ присвячено аналізу ключових компонентів, принципи їх взаємодії, а також сучасні інженерні рішення, що уможливлюють розробку масштабованої та адаптивної архітектури.

Досліджено сучасні підходи до побудови систем радіочастотного моніторингу, що дозволило окреслити їх основні завдання, зокрема виявлення спектральних аномалій, таких як стрибки амплітуди сигналу, нестабільність несучої частоти чи перекриття частот, описаних у додатку А. Встановлено, що апаратні засоби, зокрема мікроконтролери ESP32 та одноплатні комп'ютери Raspberry Pi 4, є оптимальними для реалізації таких систем завдяки їхній продуктивності, енергоефективності та гнучкості програмування.

Значну увагу приділено програмній архітектурі, де протокол MQTT забезпечує швидку передачу даних у реальному часі, а середовище Node-RED полегшує інтеграцію з вебінтерфейсами для зручної взаємодії з користувачем. Розглянуто методи обробки критичних подій, валідації команд і забезпечення стійкості системи до перешкод, включаючи резервування та захист доступу в умовах втрати зв'язку. Логічна структура системи, що включає модулі збору, обробки та передачі даних, підтверджує її масштабованість і адаптивність до динамічного радіочастотного середовища. На основі проведеного аналізу сформовано технічне завдання, яке враховує вимоги до апаратного й програмного забезпечення та специфіку роботи в умовах радіоелектронних завад. Отримані результати створюють міцне підґрунтя для подальшого проектування, моделювання та тестування системи, що буде розглянуто в наступних розділах.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

РОЗДІЛ 2 РОЗРОБКА І РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ТА КЕРУВАННЯ

2.1 Вибір середовища розробки та інструментів

Реалізація проекту вимагає залучення двох окремих платформ з відмінними підходами до розробки. Центральний вузол системи, побудований на основі одноплатного комп'ютера Raspberry Pi 4, виконує ключові обчислювальні завдання -, обробку даних з спектроаналізатора, фільтрацію аномалій, формування реакційних сценаріїв та координацію підключених виконавчих модулів. Для програмування цієї частини використано мову Python, що є оптимальним вибором завдяки її сумісності з численними бібліотеками для роботи з MQTT, обробки сигналів і побудови інтерфейсів.

У той час як керування периферійними пристроями здійснюється на мікроконтролерах ESP32, для яких застосовується середовище Arduino IDE. Обрана мова – C++ з Arduino-фреймворком - забезпечує низькорівневий доступ до апаратури, стабільність у роботі з периферією та широку підтримку бібліотек. Уся логіка, пов'язана з отриманням команд, керуванням виконавчими елементами та зв'язком через MQTT, реалізована саме на цій платформі.

Систему керування інтерфейсами побудовано за змішаним підходом. Основною візуалізаційною платформою є Node-RED, що працює на Raspberry Pi і забезпечує вебінтерфейс користувача, графічне керування та обробку повідомлень. Паралельно застосовується Flask-сервер, який доповнює функціональність системи, зокрема забезпечує REST-доступ, а також дозволяє реалізувати Telegram-бота як альтернативний канал керування. Такий підхід дозволяє розділити інтерфейсні та фонові задачі, а також створити зручний користувацький досвід.

У ролі брокера повідомлень застосовано Mosquitto, що функціонує як локальний MQTT-сервер на Raspberry Pi. Його обрано за простоту налаштування, високу продуктивність та сумісність із широким набором клієнтів, зокрема Paho

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

MQTT, яким реалізовано взаємодію як із мікроконтролерами, так і з вебінтерфейсом.

Як середовище розробки для Python-частини використовувалось Thonny IDE, яке є зручним для початкової ітерації, а для ESP32 - Arduino IDE. Паралельно застосовувались базові можливості VS Code для роботи з конфігураційними файлами та аналізу журналів MQTT. У майбутньому можливе розширення функціоналу за допомогою SDK для Telegram-бота, а також бібліотек типу fastapi, numpy, matplotlib - залежно від потреб подальшої модифікації системи.

2.2 Реалізація функціональних модулів

2.2.1 Реалізація збору даних із радіоспектра

Для забезпечення первинного аналізу радіочастотного середовища у КСМ використано спектроаналізатор TinySA Ultra.



Рисунок 2.1 - Спектроаналізатор TinySA Ultra

Цей пристрій дозволяє проводити детальний спектральний аналіз у широкому діапазоні частот (від 100 кГц до 5.3 ГГц), що забезпечує достатню гнучкість для виявлення радіочастотних аномалій у динамічному середовищі.

TinySA Ultra виступає як повноцінний апаратний компонент системи. Його використання пов'язане з потребою в автономному, портативному та відносно недорогому інструменті, який може передавати дані далі до центрального вузла без необхідності втручання користувача.

Передача даних до Raspberry Pi 4 - здійснюється через USB-інтерфейс. На стороні Raspberry Pi використовується програмний модуль, написаний мовою

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

Python, який зчитує потоки даних з порту та здійснює попередню фільтрацію. Використання бібліотек pyserial та matplotlib дозволяє здійснювати як читання, так і базову візуалізацію сигналів у реальному часі, що спрощує налагодження системи.

Реалізовані функції програмного модуля включають:

- ініціалізацію порту обміну;
- зчитування спектральних значень з пристрою;
- обробку отриманих даних (усереднення, згладжування);
- додавання часових міток;
- формування структурованого пакету для подальшої обробки.

Усі дані передаються у форматі JSON для сумісності з іншими модулями системи. Структура пакету включає не лише спектр, але й метадані: час вимірювання, канал, рівень сигналу та статус роботи пристрою. Це дає змогу швидко інтегрувати інформацію в загальний канал обробки.

Згідно з тестуванням, за умови фіксованого сканування в діапазоні 2.4–2.5 ГГц з дискретністю 1 МГц, модуль здатен передавати близько 20 оновлень на секунду, що є достатнім для більшості задач оперативного моніторингу. Крім того, можливе налаштування конфігурацій пристрою через SCPI-команди, що забезпечує адаптивність у різних сценаріях розгортання.

При втраті зв'язку або нестабільного з'єднання з Raspberry Pi, передбачено механізм повторної ініціалізації порту та кешування останніх даних у буфері, що дозволяє уникнути втрат інформації. Цей механізм критично важливий для умов, де зв'язок періодично втрачається через завади чи фізичне розташування модулів.

2.2.2 Реалізація обробки даних на Raspberry Pi

З отриманням інформації з пристрою TinySA Ultra, наступним етапом є її обробка на центральному вузлі системи - Raspberry Pi 4. Цей мінікомп'ютер обрано через достатній обсяг оперативної пам'яті, підтримку сучасних інтерфейсів та можливість безперервної роботи у складі вбудованих систем.

Основні задачі, що виконує обчислювальний вузол:

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

- приймання спектральних даних через USB або інший інтерфейс;
- аналіз та фільтрація сигналів для виявлення аномалій;
- реєстрація подій у системному журналі;
- формування відповідних керуючих команд;
- взаємодія з інтерфейсами керування та виконавчими модулями через MQTT або HTTP.

Для обробки сигналів використовується мова програмування Python у середовищі Thonny, яке оптимізоване для Raspberry Pi. Це забезпечує гнучкість та зручність розробки, а також дає змогу швидко масштабувати функціонал.

Прийом даних реалізовано з використанням бібліотеки pyserial. Потік спектральних значень зчитується з порту, обробляється в режимі реального часу та передається на наступні етапи аналізу. У разі виявлення помилок (наприклад, втрата з'єднання або пошкоджені пакети), передбачено резервний механізм перезапуску з'єднання та ведення журналу збоїв.

Фільтрація та аналіз даних здійснюється через адаптивний алгоритм, який зіставляє поточну ситуацію із заздалегідь визначеними шаблонами. Наприклад, стрибки амплітуди понад встановлений поріг, частотні повтори в короткий проміжок часу або зростання спектральної енергії в заборонених ділянках розцінюються як підозрілі події.

Після виявлення аномалій генерується запис до журналу подій, що містить часову мітку, джерело даних, тип виявленої ситуації та код реакції. Формат журналу - уніфікований, у JSON-структурі, що спрощує подальший аналіз або збереження в базі даних.

Сценарне реагування - важливий компонент. Залежно від типу виявленої події, Raspberry Pi ініціює відповідні дії. Наприклад, у випадку виявлення потенційної загрози - відправка MQTT-команди для активації РЕБ-модуля або надсилення сповіщення користувачу через Telegram-бот.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Комунікація з іншими частинами системи реалізована через MQTT-брокер, що дозволяє Raspberry Pi бути центром обміну повідомленнями. Для цього використовується бібліотека `raho-mqtt`, що забезпечує стабільну інтеграцію як з ESP32, так і з вебінтерфейсами чи мобільними застосунками.

У разі втрати з'єднання з певним модулем система не припиняє роботу - замість цього активується автономний режим обробки. Дані зберігаються локально, а команди буферизуються до моменту відновлення зв'язку. Це підвищує надійність загальної архітектури та гарантує збереження інформації навіть в умовах часткової недоступності мережі.

2.2.3 Керування виконавчими модулями

Виконавчі модулі системи призначені для реалізації реактивних дій у відповідь на події, виявлені під час аналізу радіочастотного середовища. Основою для реалізації цих модулів слугує мікроконтролер ESP32, який має достатню продуктивність, вбудовані інтерфейси зв'язку (Wi-Fi, UART, SPI) і підтримку асинхронного програмування. Його застосування дозволяє побудувати автономні, керовані пристрої, здатні оперативно реагувати на команди від центрального вузла.

Кожен ESP32-модуль виконує локальні дії: керує станом підключених виконавчих пристроїв, таких як модулятори частот, ретранслятори або системи блокування каналів. Передбачено гнучку настройку, яка дозволяє змінювати частотний діапазон, тривалість дії або режим роботи засобів впливу. Команди надходять від Raspberry Pi через MQTT-брокер. Передача виконується у форматі JSON, де чітко позначені: тип дії, параметри та допустимий часовий інтервал виконання.

У мікропрограмі ESP32 реалізовано кілька рівнів перевірки команд:

- первинна перевірка структури пакету;
- контроль дозволеного режиму відповідно до часу доби, політик безпеки або настанов користувача;
- валідація повторюваності запитів, що запобігає можливому зацикленню.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Для взаємодії з користувачем ESP32 також надсилає зворотні повідомлення про стан виконання: успішна активація, відмова або попередження. Це сприяє прозорості управління та дозволяє центральному модулю своєчасно реагувати на несправності чи неочікувану поведінку.

Особливу увагу приділено енергоспоживанню та надійності. У разі втрати зв'язку з ЦОМ, мікроконтролер переходить у попередньо заданий безпечний режим або продовжує останню дію, якщо вона є критичною для безпеки. Програмна логіка ESP32 допускає автономну зміну станів при виконанні умов, які задані в прошивці - наприклад, перевищення часу без зворотного зв'язку або зміна фізичних параметрів.

Для реалізації логіки модуля використовується Arduino IDE, яка забезпечує гнучке середовище для швидкої розробки та налагодження. Комунікація через MQTT реалізована за допомогою бібліотеки PubSubClient. Встановлено часові таймери, watchdog-механізм, а також функції запису журналів у внутрішню пам'ять для подальшої діагностики.

Таким чином, ESP32-модулі виступають як автономні керуючі вузли, здатні отримувати, перевіряти та виконувати команди, забезпечуючи при цьому високий рівень стабільності, прозорості та безпеки роботи системи у реальному часі.

2.2.4 Інтеграція з вебінтерфейсом та Telegram ботом

Система охоплюватиме декілька паралельних каналів доступу: вебінтерфейс, Telegram-бот і мобільний застосунок IoT MQTT Panel. Така мультиплатформенність не лише підвищує зручність, а й створює резервні сценарії на випадок втрати доступу до одного з каналів.

Центральним елементом вебінтерфейсу є Node-RED, що працює на Raspberry Pi. Це середовище дозволяє візуалізувати дані в режимі реального часу, керувати параметрами роботи пристроїв і переглядати журнал подій. Для спрощення керування вбудовано інтерактивні елементи: перемикачі режимів, графіки спектру, кнопки активації частот.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Telegram-бот реалізований як легкий канал сповіщення і віддаленого керування. Він приймає команди у вигляді текстових запитів, реагує на ключові події (наприклад, виявлення загрози, активація РЕБ-модуля) і надсилає відповідні повідомлення користувачу. При цьому, доступ обмежений авторизованими ID користувачів, а критичні команди потребують підтвердження. Завдяки Telegram API, бот також надсилає лог-файли, скріншоти вебінтерфейсу або короткі звіти.

Мобільний застосунок IoT MQTT Panel для iOS виступає як альтернативний спосіб моніторингу. Користувач отримує оновлення через MQTT-брокер, а також може надсилати прості команди керування. Панель настраюється вручну, що дозволяє створити кастомізований інтерфейс: наприклад, окремі кнопки для кожного режиму чи індикатори статусу модулів.

Комунікація між інтерфейсами та системою ґрунтується на MQTT, що дозволяє централізовано обробляти повідомлення, розподіляти дані та уникати дублювання. Для обробки запитів із Telegram-бота створено окремий Python-сервер, який прослуховує вхідні повідомлення та формує відповідні MQTT-пакети. У вебінтерфейсі Node-RED аналогічно працює блок логіки, що приймає відповіді та публікує результат у брокер.

Система передбачає мінімальне навантаження на ресурси Raspberry Pi, оскільки всі інтерфейси працюють з обмеженою частотою оновлення, а динамічні дані кешуються локально. Це дозволяє зберігати стабільність навіть при одночасному доступі кількох користувачів.

Завдяки гнучкій інтеграції кількох каналів керування, система адаптується до вимог користувача і дозволяє безперервно стежити за станом мережевого середовища, швидко реагувати на аномалії та змінювати параметри дії в реальному часі.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

2.2.5 Забезпечення надійності, автономності та захисту дій

Стабільна робота в умовах завад, переривань зв'язку або потенційного втручання - один із критичних параметрів для систем моніторингу та керування в радіочастотному середовищі. Тому на етапі розробки було враховано кілька рівнів забезпечення надійності, автономності та захисту дій.

1. Автономність при втраті зв'язку

У разі втрати з'єднання з Raspberry Pi, допоміжні модулі (зокрема ESP32) автоматично переходять у локальний режим. У ньому зберігаються останні відомі конфігурації, ведеться тимчасовий журнал подій, а критичні функції - наприклад, вимкнення виконавчих модулів - залишаються активними. Дані тимчасово записуються у вбудовану пам'ять і передаються до центрального вузла після відновлення з'єднання.

2. Захист від некоректних або ворожих дій

Для запобігання помилковим командам та зловмисним запитам передбачено багаторівневу перевірку команд. Вона включає:

- перевірку допустимих діапазонів (наприклад, частоти для РЕБ);
- узгодженість із поточним режимом (наприклад, заборона активації РЕБ при діагностиці);
- перевірку джерела ініціації: лише авторизовані інтерфейси мають повний доступ.

Усі критичні дії можуть бути реалізовані лише після повторного підтвердження - як з боку інтерфейсу, так і з боку центрального обчислювального модуля. Такий підхід дозволяє мінімізувати ризик помилкових активацій.

3. Резервування сценаріїв

Система передбачає наявність резервних сценаріїв для кожного основного функціонального блоку. Наприклад:

- при збої в MQTT-зв'язку переходить на локальну буферизацію;
- у разі аварійного завершення роботи одного з модулів — відключає лише відповідний вузол, а не всю систему;

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

- при втраті даних - використовує останній збережений стан як базову конфігурацію.

4. Стратегія відновлення після збоїв

У разі критичних подій система формує звіт з часовими мітками, який потім може бути проаналізований. Після перезапуску вона автоматично відновлює свій стан за попередньо збереженим профілем. Це зменшує час простою та втрату даних.

5. Адаптація до зовнішніх умов

Система реагує на зміни в спектрі, завади, різкі зміни амплітуди або втрату синхронізації. При виявленні повторюваних аномалій активуються заздалегідь задані стратегії реагування - наприклад, зміна частоти моніторингу або повідомлення оператору через Telegram.

6. Безпека передачі даних

Передбачене використання JWT для авторизації інтерфейсів. Це дає змогу захистити канали від несанкціонованого доступу, навіть якщо з'єднання відбувається через публічну мережу. Також для MQTT-брокера реалізовані SSL-сертифікати та механізми обмеження доступу за IP.

2.3 Реалізація інтерфейсів керування

2.3.1 Реалізація інтерфейсу керування через вебплатформу

Організація вебінтерфейсу в комп'ютерній системі моніторингу радіочастотного середовища має забезпечити зручний, безпечний і функціональний доступ до інформації про стан пристроїв, журнал подій, а також інструменти для керування виконавчими модулями. З огляду на обмежені ресурси одноплатного комп'ютера Raspberry Pi 4, вибір технологічного стеку повинен забезпечувати баланс між швидкістю розробки, простотою обслуговування та ефективністю в роботі з локальними мережами.

У цьому проєкті реалізацію вебінтерфейсу було вирішено виконати за змішаним підходом. Основу складає Node-RED - інструмент візуального програмування,

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

який значно спрощує розробку логіки зв'язку з пристроями та керування через MQTT-брокер. Саме Node-RED відповідає за графічну побудову блок-схем логіки керування, збирання повідомлень із тем моніторингу, генерацію подій і відображення змін у системі.

Водночас для розширення функціоналу, зокрема створення REST-інтерфейсів, окремих HTML-сторінок і API для клієнтів, було використано Flask - мікрофреймворк на мові Python. Завдяки цьому підходу вдалося розділити відповідальність між фронтендом (Node-RED) і серверною логікою (Flask), не перевантажуючи систему одночасним виконанням великої кількості скриптів.

На головній сторінці вебінтерфейсу відображається стан спектра в режимі реального часу, який передається з обчислювального вузла через MQTT. Візуалізація здійснюється через Node-RED Dashboard або окремий клієнт з графіками. Користувач бачить останні події, список активних частот, статус підключення виконавчих модулів (ESP32) та отримує повідомлення про інциденти у вигляді журналу.

Функціональні можливості вебплатформи:

- Перегляд поточних параметрів спектрального середовища.
- Вмикання та вимикання РЕБ-модулів, зміна режимів (наприклад, тестовий, активний, автоматичний).
- Завантаження журналу подій, фільтрація записів за критеріями (час, тип події, джерело).
- Надсилання команд із підтвердженням - з валідацією за допомогою сервера Flask.
- Авторизація користувачів на базі токенів для розмежування прав доступу.

З міркувань безпеки Flask реалізує контроль цілісності запитів та обмеження частоти звернень до критичних функцій. Крім того, більшість MQTT-тем захищені паролем, а надсилання конфіденційних команд вимагає ручного підтвердження в інтерфейсі.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Приклад реалізації сторінки Node-RED з блоками керування наведено на [рис. 2.2](#). Всі блоки з'єднані між собою потоками даних, події обробляються асинхронно, що забезпечує миттєву реакцію системи без перевантаження мережі.

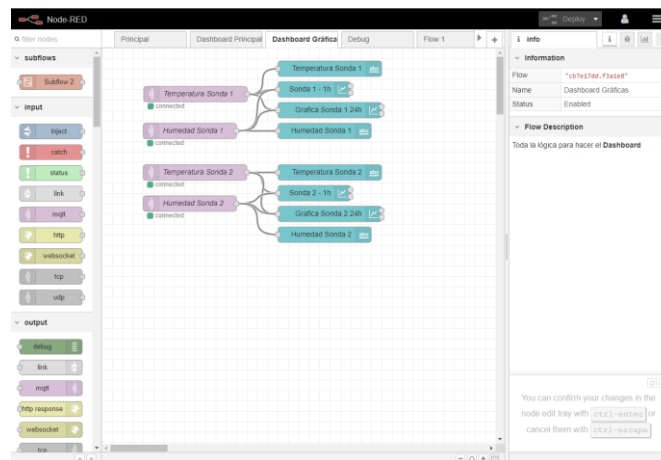


Рисунок 2.2 – Демонстрація сторінки Node-Red

2.3.2 Реалізація Telegram-бота для взаємодії з користувачем

Одним із ключових інструментів оперативної взаємодії з комп'ютерною системою є Telegram-бот. Такий формат спілкування забезпечує швидкий канал доступу до функцій системи без необхідності постійної присутності в локальній мережі або використання браузера. Окрім того, Telegram підтримує високий рівень мобільності, інформування у реальному часі та зручність використання навіть з обмеженими ресурсами з боку користувача.

У цьому проєкті Telegram-бот виконує роль асинхронного інтерфейсу, який дозволяє здійснювати як моніторинг системного стану, так і надсилання керуючих команд. Реалізацію було виконано за допомогою мови Python та бібліотеки `python-telegram-bot`, що надає всі необхідні механізми для обробки повідомлень, команд та підключення до `webhook` або `polling`-механізму.

Основні функції Telegram-бота:

- Надсилання повідомлень про виявлені аномалії в радіочастотному спектрі (наприклад, стрибки амплітуди або сигнали невідомого походження).
- Підтвердження або відхилення критичних команд (наприклад, активація РЕБ-модуля або зміна частоти).

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

- Отримання журналу подій за останній період у вигляді повідомлень або файлів.
- Запит поточного стану системи, включаючи підключення модулів, активні діапазони та статус MQTT-з'єднання.
- Можливість виконання базових керуючих дій: перезапуск вузлів, переми-
кання режимів, тестування з'єднання.

Взаємодія між Telegram-ботом та системою відбувається через проміжний сервер, розгорнутий на Raspberry Pi 4. Цей сервер фільтрує запити, виконує перевірку прав доступу (на основі списку дозволених ID користувачів), а також зв'язується з MQTT-брокером або безпосередньо з локальними скриптами, залежно від команди.

Безпека реалізації:

З огляду на можливість віддаленого керування, особливу увагу приділено контролю доступу. Всі користувачі проходять перевірку за chat_id, який порівнюється з локальним списком авторизованих клієнтів. У разі отримання запиту від невідомого користувача бот відмовляється обробляти повідомлення та інформує адміністратора про спробу доступу.

Критичні дії, як-от вмикання виконавчих пристроїв, потребують підтвердження. Для цього використовується механізм інлайн-кнопок, після натискання яких запит позначається як схвалений та запускається відповідна дія.

Типова логіка відповіді Telegram-бота:

1. Користувач надсилає команду /status.
2. Бот формує запит до обчислювального модуля.
3. Після обробки, бот надсилає стислий звіт: активні частоти, підключення модулів, час останнього інциденту.
4. Якщо активовано попередження, бот додає посилання на журнал інцидентів або пропонує підтвердити наступні дії.

Впровадження такого Telegram-інтерфейсу значно покращує оперативність реагування та зручність адміністрування системи. Завдяки мобільності користувач

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

може взаємодіяти з системою навіть у віддалених умовах, отримуючи повідомлення в реальному часі та приймаючи рішення за лічені секунди.

2.3.3 Реалізація мобільного застосунку IoT MQTT Panel для iOS

Мобільний доступ до системи моніторингу та керування є важливим елементом зручності та оперативності. У межах проєкту реалізовано мобільний інтерфейс на базі застосунку IoT MQTT Panel, що працює на пристроях з iOS. Цей застосунок забезпечує візуалізацію поточних параметрів системи, прийом повідомлень та віддалене керування виконавчими модулями через протокол MQTT.

Застосунок працює як клієнт MQTT-брокера, що розміщений на Raspberry Pi 4 або хмарному сервері. Після встановлення з App Store та базового налаштування, користувач отримує доступ до структурованої панелі управління, де кожен елемент (віджет) відповідає окремому параметру системи або дії.

Основні можливості панелі:

- Відображення статусів: активність РЕБ-модуля, частотні діапазони, наявність з'єднання, критичні події.
- Кнопки керування: ввімкнення/вимкнення модулів, зміна режимів роботи, підтвердження подій.
- Вхідні повідомлення: система може надсилати push-нотифікації через MQTT із коротким описом ситуації.
- Журнали та логи: короткі витяги журналу подій відображаються як прокручувані списки або повідомлення.

Для зручності користувача створено кілька вікон: окремо для моніторингу та для керування. Всі команди передаються по захищеному з'єднанню MQTT (з TLS), а паролі до брокера зберігаються на пристрої користувача.

Безпека реалізації:

Застосунок підтримує авторизацію на рівні MQTT-брокера. Додатково передбачено обмеження прав доступу – кожен застосунок, залежно від ролі, має власні

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

топіки для читання/запису. Це знижує ризик несанкціонованих дій у разі компрометації пристрою.

Переваги IoT MQTT Panel:

- Не потребує написання коду, налаштовується за 10–15 хвилин.
- Підтримує гнучке редагування інтерфейсу під конкретні задачі.
- Забезпечує візуальний контроль за станом системи з мінімальним споживанням ресурсів.
- Можливість роботи навіть у мобільній мережі зі слабким покриттям.

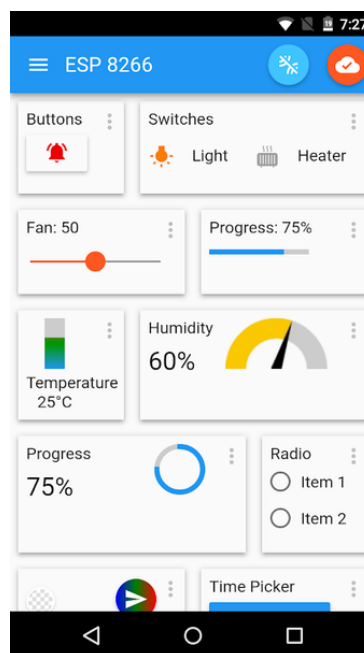


Рисунок 2.3 – Приклад інтерфейсу IoT MQTT Panel

2.4 Побудова системної логіки реагування

У центрі логіки реагування знаходиться обчислювальний модуль Raspberry Pi 4, який приймає дані зі спектроаналізатора та виконує попередню обробку. Результати проходять через серію програмних модулів, що реалізують наступні послідовні кроки:

1. Попередня фільтрація даних

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

Вхідні значення перевіряються на відповідність допустимим частотам та амплітудам. Аномальні значення (наприклад, поява сигналу у забороненому діапазоні) автоматично позначаються як потенційно критичні.

2. Класифікація подій

Використовується набір правил, згрупованих у таблиці сценаріїв. Залежно від типу події (наприклад, зростання шуму, сплеск потужності, втрата зв'язку), призначається рівень пріоритету: некритичний, помірний, критичний.

Така класифікація дає змогу системі не реагувати надмірно на фонові зміни, але залишатися чутливою до загроз.

3. Вибір дії

Для кожного типу події передбачено набір реакцій:

- Автоматичне блокування частот
- Надсилання повідомлення оператору
- Журналювання інциденту
- Активація РЕБ-модуля
- Перевірка повторюваності події протягом заданого періоду

4. Підтвердження команд

У разі критичної події, що потребує втручання, команда від оператора (через Telegram-бот або вебінтерфейс) проходить перевірку відповідності сценарію. Наприклад, якщо команда активувати глушіння надходить у період допустимого шуму, система може проігнорувати її або вимагати підтвердження другого рівня.

5. Адаптація сценаріїв

Передбачено механізм зміни поведінки системи залежно від історії подій. Наприклад, якщо протягом певного часу фіксується регулярне перевантаження одного з каналів, система оновлює допустимі пороги або переходить у розширений режим моніторингу.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Висновок другого розділу

Побудована система охоплює повний цикл роботи - від збору спектральних даних до прийняття рішень, їх передачі та зворотного контролю. Центральне значення у цьому процесі відіграє узгоджена робота обчислювальних та виконавчих пристроїв, інтерфейсів взаємодії та логіки реагування на зміни в ефірі.

Формування середовища розробки та підбір інструментів виконувалися з урахуванням обмежень апаратного забезпечення, необхідності міжплатформної взаємодії, а також з метою забезпечення стабільності й масштабованості. Використання Python та мови Arduino C дало змогу реалізувати як сценарії обробки на стороні центрального модуля, так і ефективну логіку керування для ESP32. Створена взаємодія між компонентами через MQTT-брокер забезпечує простоту обміну повідомленнями, модульність та гнучкість у модифікаціях.

Окрему увагу було приділено реалізації інтерфейсів доступу - графічних і мобільних, зокрема вебінтерфейсу, Telegram-бота та IoT-застосунку. Усі інтерфейси надають користувачеві повноцінний контроль над системою, включно з моніторингом активності, передачею команд та переглядом журналів. Така багатоканальна взаємодія не лише підвищує зручність роботи, а й сприяє адаптації системи до різних сценаріїв - автономного, операторського та гібридного.

Реалізована логіка обробки даних передбачає декілька рівнів перевірки: від базової валідації команд до детекції аномалій у спектральному середовищі. Виявлення критичних ситуацій супроводжується фіксацією події, перевіркою безпечності виконання команди та, за потреби, переходом у автономний режим. Система підтримує базові механізми захисту від перевантаження, дублювання запитів і несанкціонованих змін, що є важливими елементами стійкості.

Усі програмні модулі організовано у вигляді незалежних блоків, що взаємодіють через стандартизовані протоколи. Це спрощує подальшу модернізацію, впровадження нових функцій або адаптацію системи до іншого обладнання. Водночас

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

у системі закладено можливість масштабування - як за рахунок розширення спектрального моніторингу, так і через підключення додаткових вузлів впливу або аналітичних компонентів.

У межах реалізації були також враховані чинники надійності: збереження критичних параметрів при відключенні живлення, ведення журналу подій для постаналізу, періодична перевірка зв'язку між вузлами. Це дозволяє системі не лише підтримувати стабільну роботу, але й накопичувати історію для вдосконалення алгоритмів реагування. На практиці це означає, що система може функціонувати у динамічному середовищі без постійного нагляду, а за потреби - швидко переключатися в інші режими.

Загалом реалізована програмна частина підтвердила життєздатність обраної архітектури. Вона демонструє узгодженість функцій, ефективну взаємодію між модулями, простоту в експлуатації та відкритість до подальшого розширення. Сформоване рішення можна розглядати як базу для розгортання в прикладних сценаріях цивільного або спеціального призначення, а також як основу для більш складних інформаційно-аналітичних систем з розширеним функціоналом.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

РОЗДІЛ 3 ВИБІР ОБЛАДНАННЯ, ПОБУДОВА, НАЛАГОДЖЕННЯ ТА ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ МОНІТОРИНГУ ТА КЕ- РУВАННЯ

Цей розділ присвячений переходу від концепції до практичної реалізації комп'ютерної системи. Йдеться про створення працюючої конфігурації на основі обраних апаратних та програмних компонентів.

У межах цього етапу реалізації визначено такі ключові завдання:

- здійснити обґрунтований вибір обладнання для кожного функціонального блоку системи;
- реалізувати фізичну топологію з'єднань та відповідну логіку взаємодії між модулями;
- підготувати та налаштувати середовища для Raspberry Pi та ESP32;
- забезпечити коректну роботу локального MQTT-брокера та організувати обмін даними між підсистемами;
- перевірити стабільність з'єднань і узгодженість між протоколами UART, Wi-Fi, MQTT;
- провести тестування роботи окремих модулів та всієї системи в цілому;
- дослідити реакцію системи на збої, втрату з'єднання та аномальні ситуації.

У результаті реалізації передбачається отримати цілісну, налагоджену систему, що демонструє стійку роботу та відповідає вимогам, визначеним у технічному завданні.

3.1 Підготовка обчислювального вузла

Для реалізації обчислювального вузла в системі використано повну версію операційної системи Raspberry Pi OS з графічним середовищем. Це дозволяє не лише запускати фонові скрипти та служби, а й використовувати візуальні інструменти налаштування, моніторингу та тестування роботи модулів.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

Перед розгортанням програмного середовища було встановлено Raspberry Pi OS (повна версія з графічним інтерфейсом), що надає гнучкість під час налагодження інтерфейсів, запуску браузерних платформ, роботи з Flask та Node-RED. Установку операційної системи виконано за допомогою Raspberry Pi Imager з подальшим розгортанням образу на microSD-карту обсягом 32 ГБ класу UHS-I.

Одразу після першого запуску виконано базові налаштування:

- активовано SSH-доступ для віддаленого адміністрування;
- призначено статичну IP-адресу в межах локальної підмережі (192.168.1.150);
- встановлено оновлення системних пакетів через `sudo apt update && sudo apt upgrade -y`.

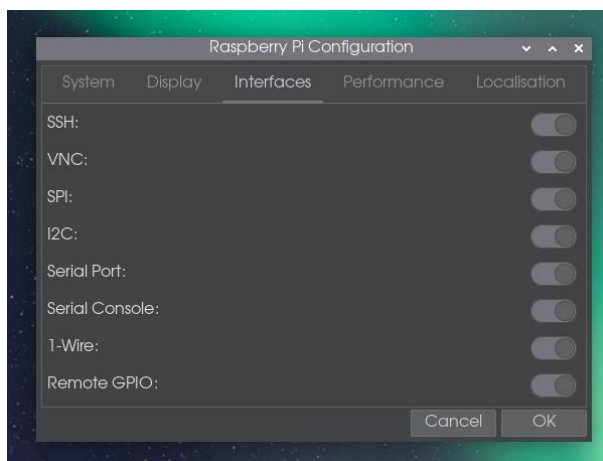


Рисунок 3.1 Налаштована конфігурація з увімкненими інтерфейсами

Також варто налаштувати VNC сервер, для цього використовуємо застосунок RealVNC. На Raspberry Pi було налаштовано порт, через який буде проводитись підключення.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

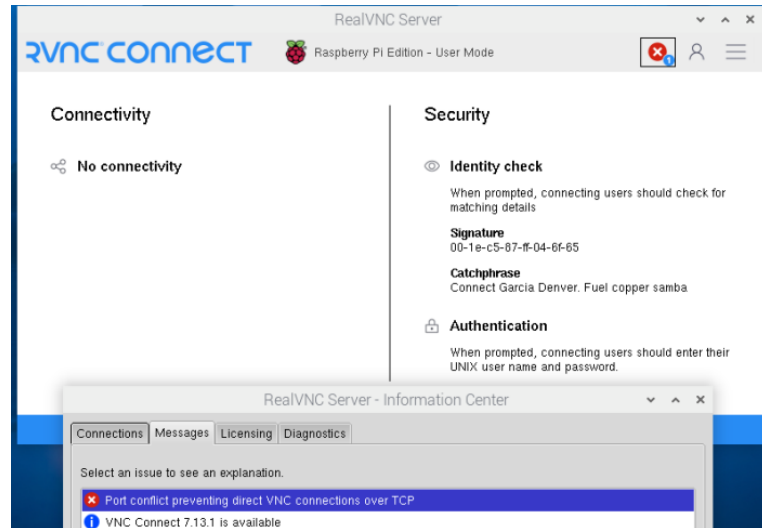


Рисунок 3.2 Головне вікно RealVNC Server

На рисунку також наведена помилка про конфлікт на порті, проте на роботу програми це не вплинуло.

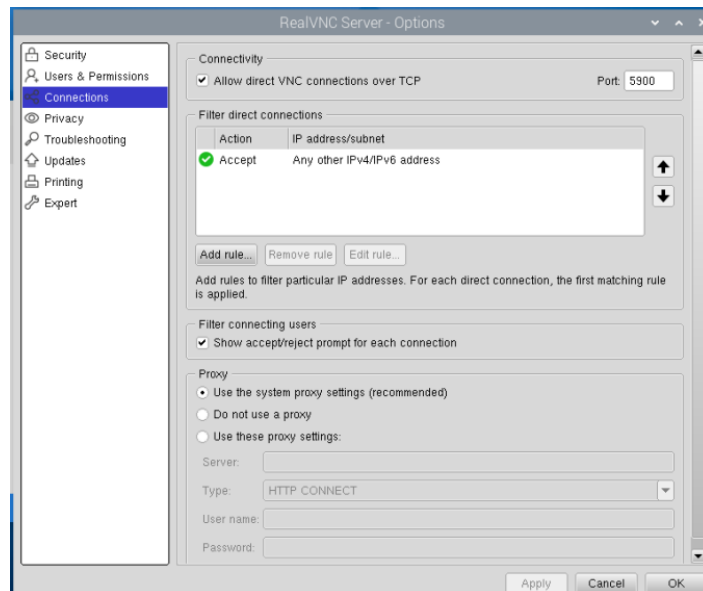


Рисунок 3.3 Вікно налаштувань, розділ в якому налаштовуємо порт підключення

Також слід подбати про підключення до сервера, для цього чудово підійде додаток RealVNC Viewer. В верхню строку вводиться IP-адреса сервера та порт для приєднання. Сервери до яких було виконано підключення відображаються в списку Address book.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

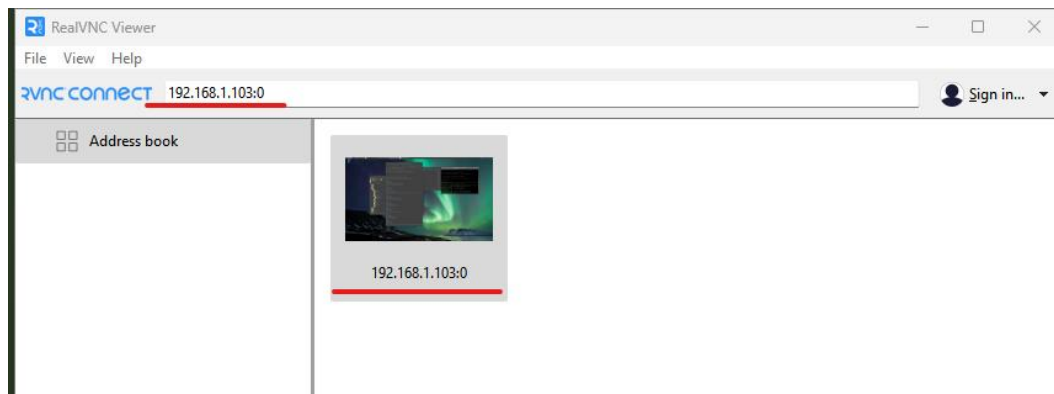


Рисунок 3.4 Програма RealVNC Viever запущена на комп'ютері

Для забезпечення стабільної взаємодії між компонентами комп'ютерної системи моніторингу та керування доцільно використовувати статичну IP-адресу для вузла на базі Raspberry Pi 4. Це дозволяє уникнути ризику зміни мережевої адреси під час перезавантаження або оновлення DHCP-конфігурацій з боку маршрутизатора, що є критичним у системах з постійним обміном даними та віддаленим доступом.

У процесі реалізації було вручну призначено IP-адресу 192.168.1.150 в межах локальної підмережі. Маску підмережі встановлено як /24, в якості шлюзу використано адресу 192.168.1.1. Для забезпечення доменного іменування та доступу до зовнішніх сервісів були вказані сервери DNS: 8.8.8.8, 1.1.1.1 та резервний локальний DNS — 192.168.1.1.

Такий підхід гарантує передбачуваність адресації, спрощує налагодження з'єднань у рамках MQTT-комунікації та вебінтерфейсу.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

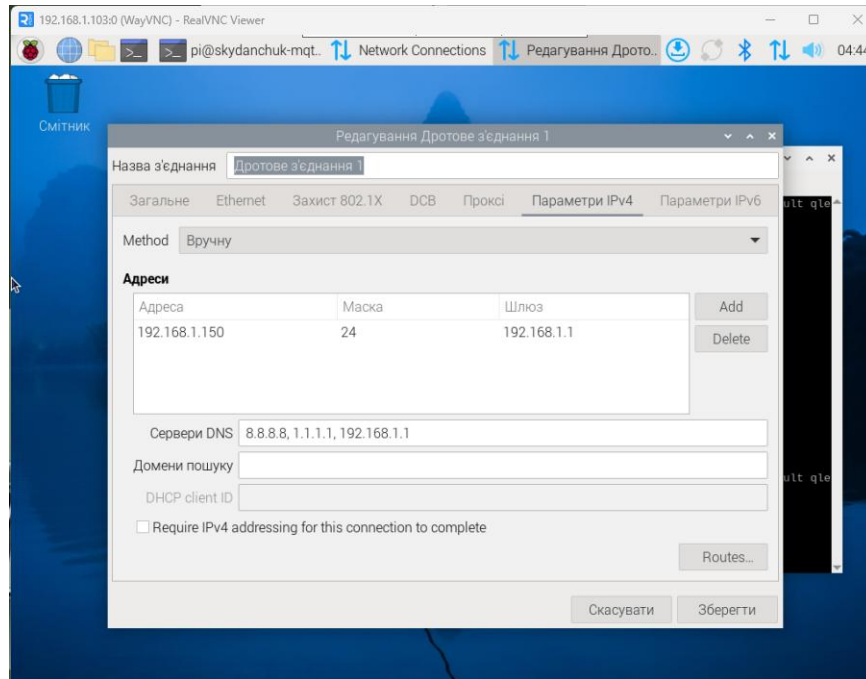


Рисунок 3.5 Вікно налаштування адреси

Наступним кроком, потрібно оновити системні файли, команда виконується в консолі Raspberry.

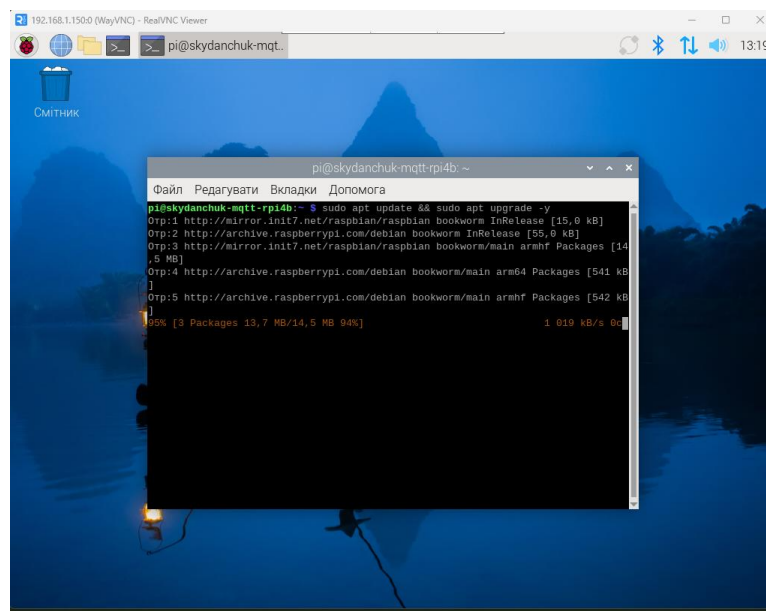


Рисунок 3.6 Робота з оновленням

3.1.1 Підготовка Python-оточення на Raspberry Pi

Наступним етапом підготовки обчислювального вузла є розгортання ізолюваного програмного середовища на базі Python, яке забезпечує запуск усіх програмних модулів системи. Вибір Python як основної мови зумовлений її сумісністю з

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

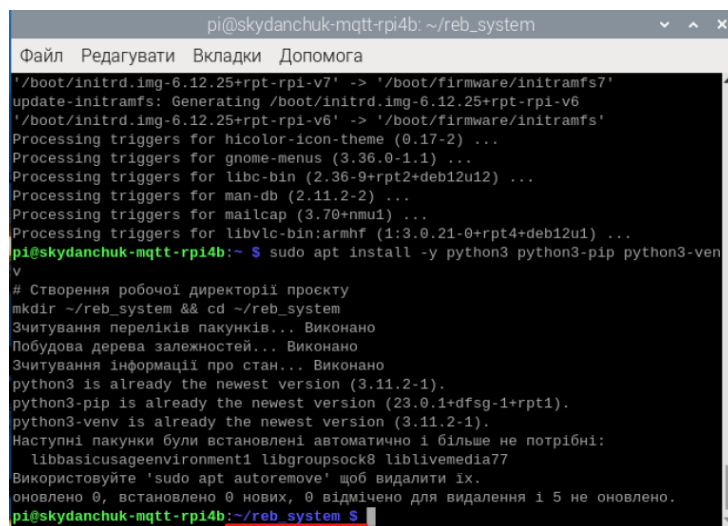
бібліотеками для роботи з MQTT, Telegram API, інтерфейсами Flask, а також високою швидкістю прототипування.

Для забезпечення стабільності та керованості залежностей було обрано підхід із використанням віртуального середовища (venv). Завдяки цьому всі бібліотеки, потрібні для системи, ізолюються від глобального оточення Raspberry Pi, що знижує ризик конфліктів між версіями модулів та спрощує супровід системи.

Етапи реалізації

Перед початком інсталяції середовища було оновлено системні пакети за допомогою стандартних команд apt. Далі створено окрему директорію проєкту `~/reb_system`, де сформовано Python-віртуальне середовище та активовано його.

На рисунку нижче показано результат виконання команд зі створення середовища та встановлення базових інструментів:



```
pi@skydanchuk-mqtt-rpi4b: ~/reb_system
Файл Редагувати Вкладки Допомога
'/boot/initrd.img-6.12.25+rpt-rpi-v7' -> '/boot/firmware/initramfs7'
update-initramfs: Generating /boot/initrd.img-6.12.25+rpt-rpi-v6
'/boot/initrd.img-6.12.25+rpt-rpi-v6' -> '/boot/firmware/initramfs'
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1.1) ...
Processing triggers for libc-bin (2.36-9+rpt2+deb12u12) ...
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for mailcap (3.70+nmui) ...
Processing triggers for libvlc-bin:armhf (1:3.0.21-0+rpt4+deb12u1) ...
pi@skydanchuk-mqtt-rpi4b:~$ sudo apt install -y python3 python3-pip python3-venv
# Створення робочої директорії проєкту
mkdir ~/reb_system && cd ~/reb_system
Зчитування переліків пакунків... Виконано
Побудова дерева залежностей... Виконано
Зчитування інформації про стан... Виконано
python3 is already the newest version (3.11.2-1).
python3-pip is already the newest version (23.0.1+dfsg-1+rpt1).
python3-venv is already the newest version (3.11.2-1).
Наступні пакунки були встановлені автоматично і більше не потрібні:
 libbasicusageenvironment1 libgroupsock8 liblivemedia77
Використовуйте 'sudo apt autoremove' щоб видалити їх.
оновлено 0, встановлено 0 нових, 0 відмічено для видалення і 5 не оновлено.
pi@skydanchuk-mqtt-rpi4b:~/reb_system$
```

Рисунок 3.7 Консоль після створення віртуального середовища Python

У межах середовища оновлено `pip` та встановлено набір основних бібліотек, зокрема:

- `raho-mqtt` — для обміну повідомленнями за протоколом MQTT;
- `flask` — для реалізації легкого вебінтерфейсу керування;
- `sqlite3` — для локального зберігання даних та логів;
- `python-telegram-bot` — для інтеграції з Telegram-ботом;
- `matplotlib` — для побудови графіків зібраних спектральних даних.

На рисунку 3.8 продемонстровано результат встановлення необхідних бібліотек через `pip`.

```
(venv) pi@skydanchuk-mqtt-rpi4b:~/reb_system $ pip install paho-mqtt flask python-telegram-bot matplotlib
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Using cached https://www.piwheels.org/simple/paho-mqtt/paho_mqtt-2.1.0-py3-none-any.whl (67 kB)
Collecting flask
  Using cached flask-3.1.1-py3-none-any.whl.metadata (3.0 kB)
Collecting python-telegram-bot
  Downloading python_telegram_bot-22.1-py3-none-any.whl.metadata (17 kB)
Collecting matplotlib
  Downloading matplotlib-3.10.3.tar.gz (34.8 MB)
----- 34.8/34.8 MB 898.0 kB/s eta 0:00:00
```

Рисунок 3.8 Встановлення бібліотек у Python-середовищі

Після завершення налаштування всі залежності було зафіксовано у файлі `requirements.txt` (переглянути вміст файлу за [додатком В](#)), що дозволяє швидко відтворити середовище на іншому пристрої за потреби. Для виходу з середовища використано команду `deactivate`.

Таким чином, підготовлене Python-оточення забезпечує платформу для подальшої розробки та запуску модулів комп’ютерної системи.

Усі команди, що були використані для створення та налаштування Python-оточення, наведено у [Додатку Б.1](#).

3.1.2 Налаштування MQTT-брокера Mosquitto

У розробленій комп’ютерній системі моніторингу та керування передбачено взаємодію між окремими модулями за допомогою протоколу MQTT. Даний протокол належить до категорії легковагових, працює за моделлю публікація–підписка, а також характеризується стабільною роботою в умовах обмеженої пропускну здатності каналу зв’язку. Такий підхід дає змогу організувати ефективний обмін повідомленнями між вузлами на базі Raspberry Pi, ESP32 та іншими пристроями, що використовуються в системі.

Для розгортання брокера обрано програму Mosquitto — перевірене у промислових і дослідницьких системах рішення з відкритим кодом, яке підтримує базові та розширені функції MQTT-протоколу, зокрема керування доступом, ведення журналів та підтримку шифрування. Брокер було встановлено безпосередньо на Raspberry Pi, що виступає в системі як центральний обчислювальний вузол.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Перед інсталяцією брокера виконано оновлення списків пакетів. Далі було встановлено програмні компоненти mosquitto та mosquitto-clients. Після завершення інсталяції служба mosquitto активується автоматично, а для забезпечення її доступності після перезавантаження використано команду активації автозапуску. На рисунку 3.9 представлено результат встановлення Mosquitto та перевірку роботи служби через системну утиліту.

```

~
lines 1-13/13 (END)...skipping...
* mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-06-26 08:53:38 EEST; 1h 35min ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Main PID: 718 (mosquitto)
    Tasks: 1 (limit: 3911)
      CPU: 1.399s
   CGroup: /system.slice/mosquitto.service
           └─718 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

чеп 26 08:53:38 skydanchuk-mqtt-rpi4b systemd[1]: Starting mosquitto.service - Mosquitto MQTT Broker.
чеп 26 08:53:38 skydanchuk-mqtt-rpi4b systemd[1]: Started mosquitto.service - Mosquitto MQTT Broker.
~

```

Рисунок 3.9 – Перевірка на працездатність

Для забезпечення мінімальної конфігурації на етапі тестування було налаштовано конфігураційний файл /etc/mosquitto/mosquitto.conf. Зокрема, вказано порт з'єднання, дозволено анонімний доступ та задано шлях до журналу подій. Такий режим роботи доцільний лише в закритому середовищі на етапі перевірки працездатності. В подальшому параметр allow_anonymous буде вимкнено, а доступ обмежено автентифікованими користувачами.

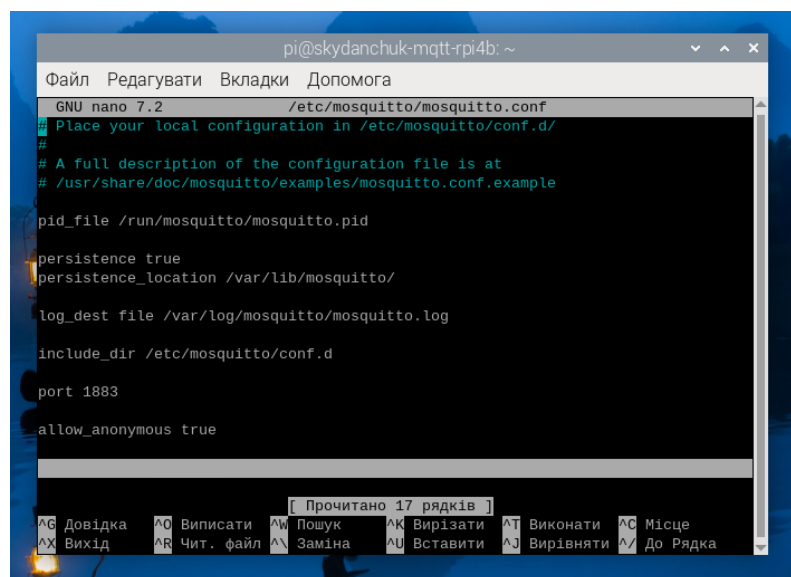


Рисунок 3.10 – Структура налаштувань після редагування.

Після завершення початкового налаштування виконано базову перевірку працездатності брокера. Для цього одночасно відкрито два термінали. В одному запущено підписку на тему test/topic, у другому — здійснено публікацію повідомлення в ту саму тему. У результаті на стороні підписника повідомлення було отримано, що підтверджує коректність встановлення і конфігурації Mosquitto-брокера. Всі дії виконувались у межах однієї машини (Raspberry Pi), що унеможливлює вплив зовнішнього середовища та спрощує діагностику.

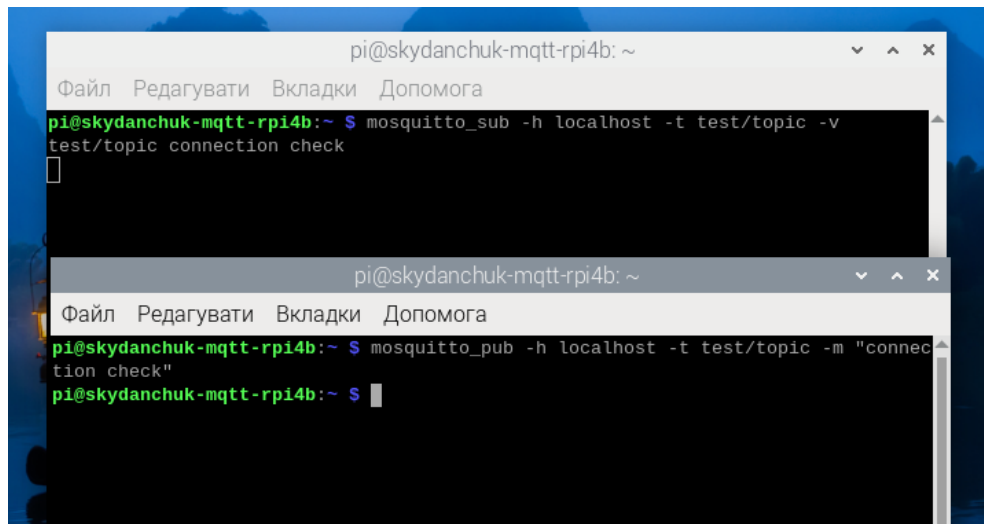


Рисунок 3.11 – Підписка на тему та отримання тестового повідомлення через локальний брокер

На даному етапі параметр allow_anonymous залишено активним винятково з метою локального тестування.

Команди, що використовувались під час встановлення, наведено у [Додатку Б.2.](#)

3.1.3 Конфігурація локальної бази даних SQLite

Для зберігання вхідних повідомлень, проміжних результатів обробки та логів було обрано вбудовану реляційну базу даних SQLite. Цей тип БД не потребує окремого серверного компонента, має низькі вимоги до ресурсів та повністю інтегрований у Python-оточення. Оскільки підтримка SQLite уже вбудована в стандартну бібліотеку Python 3, додаткового встановлення не потребувалося. Було створено

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

окремий файл бази даних reb_data.db у робочій директорії ~/reb_system, де також розміщуються інші компоненти системи.

```

pi@skydanchuk-mqtt-rpi4b: ~/reb_system
Файл Редагувати Вкладки Допомога
pi@skydanchuk-mqtt-rpi4b:~$ cd ~/reb_system
pi@skydanchuk-mqtt-rpi4b:~/reb_system$ touch reb_data.db
pi@skydanchuk-mqtt-rpi4b:~/reb_system$ ls -l
загалом 8
-rw-r--r-- 1 pi pi 0 чер 26 16:13 reb_data.db
-rw-r--r-- 1 pi pi 467 чер 26 09:05 requirements.txt
drwxr-xr-x 5 pi pi 4096 чер 26 08:22 venv
pi@skydanchuk-mqtt-rpi4b:~/reb_system$

```

Рисунок 3.12 – Створений файл бази даних у структурі проєкту

Далі за допомогою Python-скрипта сформовано початкову структуру таблиці, що містить чотири поля: унікальний ідентифікатор, мітку часу, тему повідомлення та його вміст.

```

1 import sqlite3
2
3 # Підключення до локальної бази даних (файл буде створено автоматично, якщо його ще немає)
4 conn = sqlite3.connect("reb_data.db")
5 cursor = conn.cursor()
6
7 # Створення таблиці messages,
8 cursor.execute("""
9 CREATE TABLE IF NOT EXISTS messages (
10 id INTEGER PRIMARY KEY AUTOINCREMENT,
11 timestamp TEXT NOT NULL,
12 topic TEXT NOT NULL,
13 payload TEXT NOT NULL
14 )
15 """)
16
17 # Збереження змін і закриття з'єднання
18 conn.commit()
19 conn.close()
20

```

Рисунок 3.13 – Запуск Python-інтерпретатора та створення таблиці messages

Для підтвердження успішного створення структури таблиці використано утиліту sqlite3, яка дозволяє переглядати вміст та схему БД безпосередньо з терміналу. За допомогою команд .tables та .schema messages перевірено наявність таблиці messages та її структуру.

```

pi@skydanchuk-mqtt-rpi4b: ~/reb_system
Файл Редагувати Вкладки Допомога
pi@skydanchuk-mqtt-rpi4b:~ $ cd ~/reb_system
pi@skydanchuk-mqtt-rpi4b:~/reb_system $ touch reb_data.db
pi@skydanchuk-mqtt-rpi4b:~/reb_system $ ls -l
загалом 8
-rw-r--r-- 1 pi pi  0 чеп 26 16:13 reb_data.db
-rw-r--r-- 1 pi pi 467 чеп 26 09:05 requirements.txt
drwxr-xr-x 5 pi pi 4096 чеп 26 08:22 venv
pi@skydanchuk-mqtt-rpi4b:~/reb_system $ sqlite3 reb_data.db
SQLite version 3.40.1 2022-12-28 14:03:47
Enter ".help" for usage hints.
sqlite> .tables
messages
sqlite> .schema messages
CREATE TABLE messages (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  timestamp TEXT NOT NULL,
  topic TEXT NOT NULL,
  payload TEXT NOT NULL
);
sqlite> .exit
pi@skydanchuk-mqtt-rpi4b:~/reb_system $

```

Рисунок 3.14 – Перевірка структури таблиці messages у SQLite через термінал

Створена база даних буде використовуватись у подальших розділах системи для збереження зібраної телеметрії, повідомлень про події, а також для забезпечення локального архівування у випадку тимчасової втрати зв'язку з інтерфейсами керування.

Таблиця 3.1

Структура таблиці повідомлень reb_data.db

Назва поля	Тип даних	Призначення
id	INTEGER PRIMARY KEY AUTOINCREMENT	Унікальний іденти- фікатор повідомлення
timestamp	TEXT	Мітка часу фіксації повідомлення
topic	TEXT	Тема MQTT, за якою було отримано по- відомлення
payload	TEXT	Тіло повідомлення (дані або текст події)

Усі кроки створення та ініціалізації бази даних SQLite реалізовано згідно з [Додатком Б.3.](#)

3.2 Підключення спектроаналізатора TinySA Ultra

Задля забезпечення збору спектральних даних у системі моніторингу використано портативний спектроаналізатор TinySA Ultra, який підтримує керування та отримання даних через USB-інтерфейс. У проєкті пристрій використовується як автономне джерело радіочастотної інформації в діапазоні до 6 ГГц. Його підключення здійснюється до Raspberry Pi 4 за допомогою стандартного USB-кабелю.

TinySA Ultra працює як віртуальний COM-порт і підтримує передачу даних за допомогою простого послідовного протоколу. Для обміну повідомленнями та запитів до пристрою застосовується Python-бібліотека pyserial, яка надає зручний інтерфейс для роботи з послідовним портом у рамках сценаріїв системи. Завдяки цьому реалізовано програмне отримання спектрів у текстовому форматі та подальшу передачу їх до модуля обробки.

Після фізичного підключення пристрою TinySA Ultra до Raspberry Pi через USB, його розпізнає операційна система як віртуальний послідовний порт (наприклад, /dev/ttyACM0). На [рисунку 3.15](#) наведено результат роботи програми, також було виконано фізичне представлення проєкту, що можна прослідкувати за [рисунком 3.16](#).

Програмний алгоритм, що реалізований мовою Python, у кожному циклі виконує послідовні етапи:

Ініціалізація та відкриття послідовного порту:

За допомогою бібліотеки pyserial відкривається з'єднання з аналізатором, встановлюються швидкість та таймаут обміну.

Формування та відправлення команди сканування:

Генерується команда (наприклад, scanraw із заданим діапазоном частот та кількістю точок), яка надсилається до пристрою через послідовний порт.

Отримання та збереження масиву спектральних даних:

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Після виконання сканування пристрій повертає масив бінарних значень, які приймаються програмою, очищаються від службової інформації та розпаковуються у набір амплітуд для кожної точки спектра.

Обробка та аналіз спектра:

Розпаковані дані аналізуються у реальному часі. Виконується перевірка на наявність аномалій — зокрема, виявлення перевищення порогового рівня сигналу у визначених критичних діапазонах частот.

Логуювання та реагування:

Результати вимірювань і всі ключові події (наприклад, виявлені аномалії або штатна відсутність подій) фіксуються у локальній базі даних для аудиту. При виявленні критичних подій передбачено подальше реагування — активація виконавчих модулів чи відправлення сповіщення оператору. ([формування запиту в додатку 6.4](#))

Такий підхід забезпечує гнучкість, масштабованість і повну автономність системи збору та обробки радіочастотної інформації.

```
Команда відправлена, очікуємо відповідь...
Отримано 7996 байт.
!!! ВИЯВЛЕНО АНОМАЛІЇ !!!
Частота: 432.02 МГц, Рівень: 36984
Частота: 433.01 МГц, Рівень: 30729
Частота: 434.98 МГц, Рівень: 24696
Частота: 866.37 МГц, Рівень: 30731
Частота: 869.34 МГц, Рівень: 30731
Частота: 913.87 МГц, Рівень: 30731
Частота: 915.85 МГц, Рівень: 30328
Частота: 916.84 МГц, Рівень: 30732
Частота: 2798.72 МГц, Рівень: 30728
Частота: 2800.70 МГц, Рівень: 42872
Частота: 2801.69 МГц, Рівень: 30728
```

Рисунок 3.15 – Підтвердження працездатності алгоритму

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

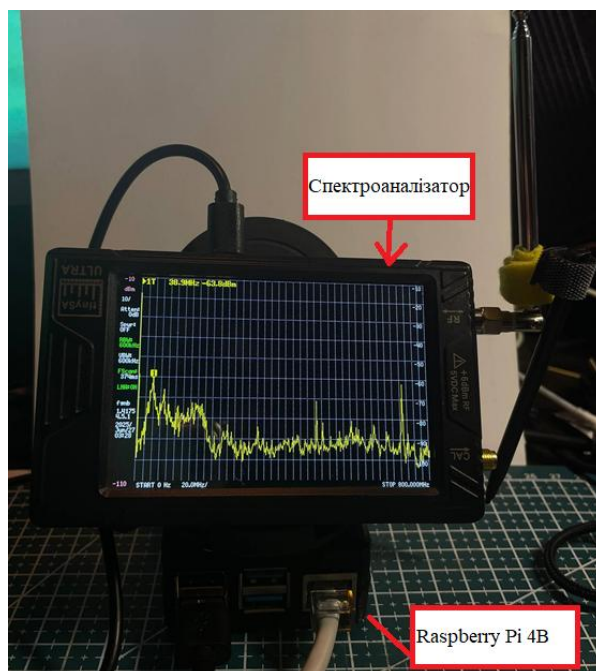


Рисунок 3.16 – Фізичне представлення частини по прийому сигналу
З повним алгоритмом можна ознайомитись за [Додатком Б.4.](#)

3.3 Налаштування взаємодії між вузлами

У системі моніторингу важливо забезпечити чіткий обмін даними між Raspberry Pi та мікроконтролером ESP32. Для цього використано протокол MQTT. Його реалізація дозволяє швидко передавати команди та отримувати відповіді у режимі реального часу.

Обидва вузли підтримують загальний простір повідомлень. Це дає змогу централізовано збирати дані та керувати виконавчими модулями. Такий підхід підвищує стабільність системи, спрощує масштабування та адаптацію до нових задач. У цьому підрозділі розглядаються способи реалізації обміну, структура повідомлень і основні сценарії тестування взаємодії.

3.3.1 Конфігурація MQTT-клієнта на Raspberry Pi

З огляду на архітектуру системи моніторингу, основою міжвузлової взаємодії обрано протокол MQTT (Message Queuing Telemetry Transport), який дозволяє здійснювати ефективну доставку команд і подій між окремими пристроями в умовах

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

обмеженого навантаження на мережу. MQTT особливо придатний для IoT-систем завдяки своїй легкості, надійності доставки та простоті реалізації.

У проєктованій системі роль центрального обчислювального вузла виконує Raspberry Pi 4, на якому реалізовано програмну обробку команд, збір спектральних даних із пристрою TinySA Ultra, а також фіксацію результатів у базі даних. Саме Raspberry Pi функціонує як MQTT-клієнт, що підписується на топіки та реагує на вхідні сигнали з інших вузлів, зокрема з ESP32, який є передавачем команд.

Основні елементи конфігурації:

- MQTT-брокер: Mosquitto, розгорнутий локально на Raspberry Pi (IP-адреса пристрою: 192.168.1.150).
- Бібліотека клієнта: paho-mqtt для Python 3.
- Порт зв'язку: 1883 (за замовчуванням).
- Топік підписки: reb/cmd — канал отримання керуючих команд.
- Топік публікації: у межах даного проєкту не використовується.
- Механізм обробки: функціональні колбеки на подію підключення (on_connect) та отримання повідомлення (on_message).
- Формат команд: текстові запити, що оброблюються як логічні тригери (наприклад, scan).

Програма на Raspberry Pi ініціалізується автоматично під час запуску ОС та переходить у режим очікування команди від ESP32. Після надходження керуючої команди (наприклад, "scan"), клієнт здійснює зчитування спектрального діапазону, проводить аналіз отриманих даних, логування аномалій та фіксацію записів у таблиці бази даних spectrum, anomalies та messages.

Нижче наведено фрагмент функціональної частини реалізації MQTT-клієнта:

```
import paho.mqtt.client as mqtt

from Main_script import collect_spectrum, parse_spectrum,
detect_anomalies, log_to_db, log_message

DB_PATH = "/home/pi/reb_system/reb_data.db"
```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

```

def on_message(client, userdata, msg):
    command = msg.payload.decode().strip()
    print(f"[MQTT] Отримано команду: {command}")
    log_message(DB_PATH, msg.topic, command)
    if command.lower() == "scan":
        raw = collect_spectrum()
        values = parse_spectrum(raw, 2648)
        anomalies = detect_anomalies(values, ...)
        log_to_db(DB_PATH, int(time.time()), values, anomalies)

def on_connect(client, userdata, flags, rc):
    client.subscribe("reb/cmd")

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost", 1883, 60)
client.loop_forever()

```

Цей скрипт забезпечує повний цикл взаємодії між брокером MQTT та функціями збору спектральних даних. Для роботи з MQTT у Python використано бібліотеку `paho-mqtt`, яка забезпечує інтуїтивно зрозумілий API для клієнтських підключень. Команда "scan" виступає ініціатором запуску аналізу в діапазоні від 340 МГц до 2960 МГц із кількістю точок 2648. Далі дані розпаковуються, аналізуються на предмет аномалій, а потім — зберігаються в базі даних SQLite, яка фізично розміщується в каталозі `/home/pi/reb_system/reb_data.db`.

Усі події, включаючи отримання команд, помилки з'єднання або результати аналізу, фіксуються у таблиці `messages`. Цей механізм реалізовано через допоміжну функцію `log_message(...)`.

Фізично Raspberry Pi отримує доступ до мережі через Ethernet-з'єднання з адресою 192.168.1.150, що дозволяє іншим пристроям у локальній мережі, зокрема ESP32, здійснювати публікацію команд на MQTT-брокер. Адресація підтверджена інформацією з інтерфейсу `eth0`, де налаштовано шлюз 192.168.1.1, що забезпечує доступ до мережі та внутрішню маршрутизацію.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

На рисунку 3.17 показано процес запуску MQTT-клієнта та успішне отримання команд у терміналі Raspberry Pi

```
[MQTT] Підписано на топик: reb/cmd
[MQTT] Отримано команду: scan
Команда відправлена, очікуємо відповідь...
Отримано 7996 байт.
[MQTT] Отримано команду: scan
Команда відправлена, очікуємо відповідь...
Отримано 7996 байт.
I
```

Рисунок 3.17 – Журнал підключення MQTT-клієнта Raspberry Pi та обробка команд

Код повної реалізації клієнта MQTT, обробника спектру та функцій логування винесено до Додатку Б.5, де наведено всі необхідні компоненти проекту.

Повний перелік команд та реалізація клієнта подано у [Додатку Б.5](#).

3.3.2 Налаштування MQTT-клієнта на ESP32

Для забезпечення взаємодії вузлів системи з MQTT-брокером, що функціонує на Raspberry Pi, на мікроконтролері ESP32 реалізовано клієнтський MQTT-модуль. З огляду на обмеження в ресурсах та необхідність забезпечення стабільного бездротового зв'язку, ESP32 підключається до локальної Wi-Fi-мережі та взаємодіє з брокером Mosquitto через протокол MQTT у режимі підписки на керуючі команди.

У реалізації використано Arduino IDE як середовище розробки, а також бібліотеки WiFi.h для налаштування мережевого з'єднання та PubSubClient.h для MQTT-зв'язку.

Програмна реалізація клієнта

Після ввімкнення ESP32 підключається до визначеної точки доступу Wi-Fi. Після успішного з'єднання ініціалізується MQTT-клієнт та здійснюється підписка на топик reb/control. При отриманні повідомлення викликається функція зворотного виклику, у якій здійснюється аналіз вхідної команди. Наразі реалізовано обробку команди scan, яка моделює запуск процедури сканування спектру.

Перевірка роботи клієнта

Для тестування працездатності клієнта використано інструменти Mosquitto CLI (mosquitto_pub) на стороні Raspberry Pi. Після запуску ESP32 з відкритим монітором порту в Arduino IDE, було опубліковано тестове повідомлення:

```
mosquitto_pub -h 192.168.1.150 -t "reb/control" -m "scan"
```

У результаті на ESP32 було зафіксовано повідомлення у серійному моніторі:

```
09:53:01.301 -> [MQTT] Отримано повідомлення в топіку: reb/control
09:53:01.301 -> [MQTT] Тіло повідомлення: scan
09:53:01.301 -> Отримано команду scan — імітація запуску сканування...
09:53:46.082 -> [MQTT] Отримано повідомлення в топіку: reb/control
09:53:46.082 -> [MQTT] Тіло повідомлення: scan
09:53:46.082 -> Отримано команду scan — імітація запуску сканування...
09:54:37.553 -> [MQTT] Отримано повідомлення в топіку: reb/control
09:54:37.553 -> [MQTT] Тіло повідомлення: scan
09:54:37.588 -> Отримано команду scan — імітація запуску сканування...
```

Рисунок 3.19 - Вивід повідомлення про прийом MQTT-команди на ESP32

Це підтверджує, що підписка, з'єднання та обробка команд працюють коректно.

У ході експериментів встановлено, що ESP32, підключений через Wi-Fi до тієї ж локальної мережі, у якій функціонує Raspberry Pi з Mosquitto, коректно отримує команди та здійснює відповідну обробку.

Ознайомитись з лістингом алгоритму налаштувань можна у [Додатку Б6](#)

3.4 Telegram-бот: розгортання, інтеграція, сценарії використання

Для зручної взаємодії з користувачем у системі реалізовано Telegram-бота, який забезпечує базову індикацію стану, контроль обміну повідомленнями та можливість керування виконавчими модулями. Комунікація відбувається через офіційне API Telegram за допомогою бібліотеки python-telegram-bot.

Розгортання Telegram-бота виконується на Raspberry Pi, де вже функціонує брокер MQTT. Бот запускається як окремий модуль у фоновому режимі та прослуховує текстові команди користувача. При цьому зберігається гнучкість завдяки використанню асинхронної моделі обробки подій.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

У системі передбачено набір команд, які реалізують базову діагностику та можливість ручного втручання в роботу системи. Таблиця нижче подає можливості бота

Таблиця 3.2

Функціональні можливості бота

Команда	Опис
/start	Привітальне повідомлення, підтвердження готовності бота до роботи.
/status	Показ поточного статусу системи моніторингу.
/help	Вивід переліку доступних команд з короткими поясненнями.
/last	Вивід п'яти останніх зафіксованих аномалій із частотою та рівнем сигналу
/dbsize	Показ розміру локальної бази даних аномалій у кілобайтах.
/cmd	Передача довільної MQTT-команди в топик reb/control (наприклад, /cmd scan)
/activate	Активує виконавчий пристрій (умовно — РЕБ) на заданому частотному діапазоні.

Особливу увагу слід звернути на реалізацію команди /activate, яка дозволяє активувати систему у визначеному частотному діапазоні. Передбачено перевірку допустимого діапазону частот (450–2450 МГц), після чого бот формує MQTT-повідомлення у форматі:

makefile

КопіюватиРедагувати

activate:<частота>

та публікує його у топіку reb/control, на який підписано мікроконтролер ESP32. Це дозволяє реалізувати пряму взаємодію між користувачем та виконавчим пристроєм, наприклад, увімкнення індикатора чи запуск певного сценарію.

Приклад використання:

Користувач надсилає команду:

```
/activate 915
```

У відповідь ESP32 приймає MQTT-повідомлення activate:915, інтерпретує його як вказівку на активацію системи в діапазоні 450–2450 МГц і вмикає відповідний пін, до якого може бути підключено світлодіод чи інший виконавчий пристрій.

Весь код Telegram-бота реалізовано у вигляді окремого Python-модуля з інтеграцією доступу до SQLite-бази даних (reb_data.db). Для передачі MQTT-повідомлень використовується бібліотека paho-mqtt, що дозволяє надсилати керуючі пакети безпосередньо з Raspberry Pi.

У якості прикладу, на рисунку нижче показано фрагмент роботи Telegram-бота з обробкою команд /status, /last та /activate:

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

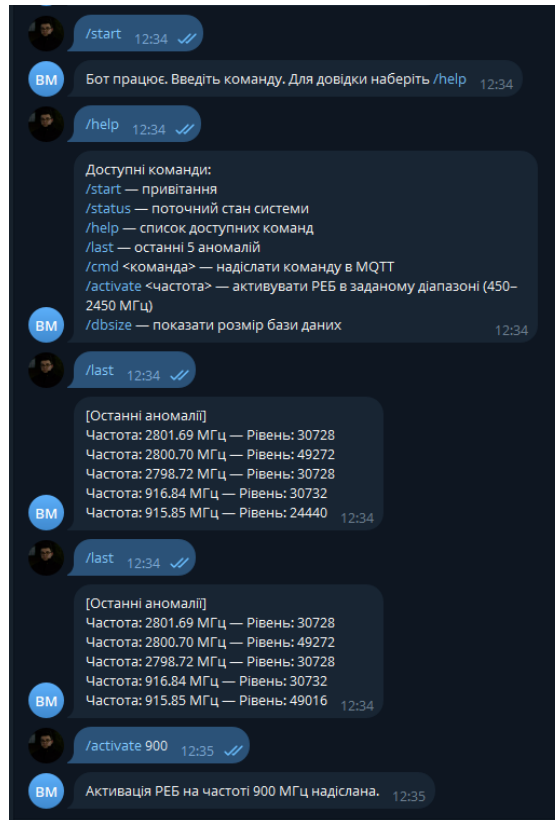


Рис. 3.20 – Взаємодія користувача з Telegram-ботом у процесі керування

Взаємодія відбувається в реальному часі, що дозволяє швидко реагувати на критичні події у спектрі або вручну ініціювати перевірку системи. Також реалізовано функції захисту від помилкового введення (наприклад, невірною формату команди або частоти поза допустимим діапазоном).

Таким чином, Telegram-бот виступає не лише засобом індикації, а й ефективним інтерфейсом віддаленого контролю системи.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

Висновок третього розділу

У межах третього розділу було здійснено практичне розгортання, налаштування та тестування спроектованої комп'ютерної системи. Сформований у попередніх етапах концепт системи набув конкретної технічної реалізації, що охоплює інтеграцію апаратної платформи, програмних модулів, засобів взаємодії та обробки даних.

Основна увага була приділена створенню стійкої та масштабованої інфраструктури, що забезпечує безперервне отримання спектральної інформації, її локальне збереження, обробку та передачу до інтерфейсів сповіщення і реагування. Розгортання ключових компонентів – зокрема MQTT-брокера, локальної бази даних, модулів збору та аналізу даних – було виконано з урахуванням вимог до надійності, низької затримки та відповідності апаратним обмеженням.

Завдяки використанню мікрокомп'ютера Raspberry Pi як центрального вузла обробки вдалося реалізувати повноцінний контроль за поточним спектральним середовищем у заданих діапазонах. Зафіксовані аномалії передаються до логічного ядра, де здійснюється їх перевірка та запуск механізмів реагування. Наявність окремого виконавчого модуля, зібраного на базі ESP32, дала змогу відокремити етап прийняття рішення від апаратної дії, що позитивно впливає на структурованість і безпеку системи.

Реалізація Telegram-бота для оперативного сповіщення та надсилання команд, а також мобільного MQTT-клієнта для ручного контролю і моніторингу. Це забезпечило зручність взаємодії з системою навіть для користувачів без спеціальної підготовки, а також підвищило прозорість процесів, що відбуваються в системі. Передбачено інструменти перегляду останніх аномалій, аналізу журналу повідомлень, перевірки стану бази даних, а також можливість ініціювання реакцій через прості текстові команди.

Окрему увагу було приділено тестуванню працездатності системи в умовах, наближених до реальних сценаріїв використання. Здійснено перевірку обміну по-

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

відомленнями між вузлами, імітацію виявлення критичних частот, а також моделювання керування виконавчим пристроєм. Виявлені у ході випробувань затримки, збої або конфлікти були локалізовані та усунені, що підтвердило життєздатність обраної архітектури та правильність розроблених алгоритмів.

У результаті реалізації технічної частини проєкту було досягнуто балансу між функціональністю, стабільністю та простотою впровадження. Розгортання системи не потребує складної конфігурації або використання закритого програмного забезпечення, що відкриває можливості для подальшої адаптації та масштабування рішення в рамках інших інфраструктур або розширених сценаріїв радіомоніторингу.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

ВИСНОВКИ

У межах кваліфікаційної роботи, присвяченої проєктуванню комп'ютерної системи управління комплексом радіоелектронної боротьби та розвідки, реалізовано всі поставлені завдання. За підсумками аналізу можна стверджувати, що результати відповідають як формальним вимогам, так і практичній цінності розробки. Створене рішення є цілісним і може бути основою для подальшого вдосконалення у сфері технічних систем захисту та моніторингу.

У першому розділі було обґрунтовано доцільність створення систем моніторингу радіочастотного середовища. Визначено сучасні підходи до побудови таких систем, здійснено аналіз апаратної бази, а також обрано платформи ESP32 та Raspberry Pi 4 як основні обчислювальні вузли. Окрему увагу приділено методам виявлення спектральних аномалій, що мають практичне застосування у сфері радіотехнічної розвідки. Встановлено технічні вимоги до реалізації системи та побудовано її логічну архітектуру. Ці теоретичні напрацювання стали основою для реалізації практичної частини.

Другий розділ було присвячено реалізації функціональних компонентів системи. Успішно впроваджено модулі збору спектральних даних, обробки результатів, а також взаємодії з користувачем через вебінтерфейс та Telegram-бота. Окремо описано механізми забезпечення надійності та роботи в умовах автономного режиму. Впроваджена логіка реагування дозволяє здійснювати контроль і керування у режимі реального часу. Завдяки структурованій побудові інтерфейсів забезпечено зручність користування та оперативність у виконанні команд. Цей етап став ключовим у створенні працездатного прототипу.

Третій розділ охопив вибір обладнання, конфігурацію мережевих сервісів та тестування розробленої системи. Вдалося реалізувати взаємодію між вузлами на основі MQTT-протоколу, налаштувати брокер Mosquitto, забезпечити зберігання даних у SQLite та інтегрувати спектроаналізатор TinySA Ultra. У процесі тестування підтверджено синхронність дій між вузлами та стабільність обміну повідом-

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

леннями. Результати випробувань у локальному середовищі показали високу ефективність запропонованої архітектури. Система показала стійкість до помилок та здатність до адаптації в умовах змін середовища.

Отже, поставлені на початку роботи завдання виконані. Отримані результати не лише демонструють ефективність запропонованого рішення, а й створюють передумови для подальшого розвитку системи. Зокрема, перспективними напрямками є розширення функціоналу, інтеграція додаткових сенсорів, а також оптимізація обробки даних у режимі реального часу.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Матеріали I міжнародної науково-технічної конференції: “Системи і технології зв’язку, інформатизації та кібербезпеки: актуальні питання і тенденції розвитку” - Київ: ВІТІ ім. Героїв Крут, 2021. - 321 с. [Електронний ресурс] - Режим доступу:
https://sprotyvg7.com.ua/wp-content/uploads/2023/09/c_2021.pdf
2. Дисертація: “Cognitive Radio. An Integrated Agent Architecture for Software Defined Radio” by Joseph Mitola III - Sweden: Royal Institute of Technology, 2000. - 313 р. [Електронний ресурс] - Режим доступу:
<https://scispace.com/pdf/cognitive-radio-an-integrated-agent-architecture-for-1v855bviue.pdf>
3. Технічна документація: “ESP32 Technical Manual” by Espressif Systems. [Електронний ресурс] – Режим доступу:
<https://www.espressif.com/en/support/documents/technical-documents>
4. Технічна документація: “STM32H5 series - PDF Documentation” by STMicroelectronics. [Електронний ресурс] – Режим доступу:
<https://www.st.com/en/microcontrollers-microprocessors/stm32h5-series/documentation.html>
5. Технічна документація: “Raspberry Pi Documentation” by Raspberry Pi Trading Ltd, 2020. [Електронний ресурс] – Режим доступу:
<https://www.raspberrypi.com/documentation/>
6. Технічна документація: “MQTT Version 3.1.1 Plus Errata 01” by OASIS Standard, 2014. [Електронний ресурс] – Режим доступу:
<https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>
7. Книга: “Cognitive Dynamic Systems: Perception-action Cycle, Radar and Radio.” by Simon Haykin – Cambridge University Press, 2012. – 384 р [Електронний ресурс] – Режим доступу:
<https://www.cambridge.org/core/books/abs/cognitive-dynamic-systems/perceptionaction-cycle/805079DE41370F4F1003E63B1476E74D>

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

8. Технічна документація: “Node-RED Documentation” [Електронний ресурс] – Режим доступу: <https://nodered.org/docs/>
9. Технічна документація: “Python Telegram Bot Library.” [Електронний ресурс] – Режим доступу: <https://github.com/python-telegram-bot/python-telegram-bot>
10. Технічна документація: “TinySA Ultra — Technical Specifications and Community Wiki.” [Електронний ресурс] – Режим доступу: <https://www.tinysa.org/wiki/>
11. Технічна документація: “SON Web Tokens (JWT) — Introduction and Documentation” [Електронний ресурс] – Режим доступу: <https://jwt.io/introduction>

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

ДОДАТКИ

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

Класифікація радіочастотних аномалій

У вступі було згадано про аномалії в радіоефірі.

1. Стрибки амплітуди сигналу - раптове та нетипове зростання рівня потужності сигналу в окремому частотному діапазоні, що може свідчити про появу нового джерела випромінювання або активізацію передавача з високою енергетичною віддачею.
2. Поява нехарактерних сигналів - фіксація радіочастотної активності, яка відсутня у стандартному фоні, що може бути ознакою використання нестандартних засобів зв'язку або ворожої активності в ефірі.
3. Нестабільність несучої частоти - відхилення або коливання частоти сигналу від її типового значення, часто зумовлені спробами уникнення детекції або використанням дешевих або нестійких передавачів.
4. Перекриття частот - одночасна присутність кількох сигналів у межах одного вузького діапазону, що ускладнює ідентифікацію джерел та може свідчити про спробу навмисного глушіння або перевантаження спектра.
5. Імпульсні або періодичні збурення - короткотривалі високочастотні імпульси, які повторюються або виникають випадково, часто пов'язані з цифровими протоколами передачі даних або завадовими системами.
6. Зміна шумового фону - нетипове підвищення загального рівня шуму в спектрі, яке не пов'язане з відомими передавачами й може бути наслідком широкосмугового випромінювання, електромагнітної інтерференції або роботи глушильних пристроїв.

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинги програмного коду реалізації комп'ютерної системи

- [Б.1 - Підготовка Python-оточення](#)
- [Б.2 - MQTT-брокер і клієнти](#)
- [Б.3 - Робота з базою даних SQLite](#)
- [Б.4 - Налаштування логування та зчитування частоти з TinySA Ultra](#)
- [Б.5 - Зв'язок між ЦОМ та ESP 32](#)
- [Б.6 - Telegram-бот і вебінтерфейс](#)

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

Підготовка Python-оточення

```
# Оновлення списків пакетів
sudo apt update

# Оновлення встановлених пакетів
sudo apt upgrade -y

# Встановлення Python3, pip та venv
sudo apt install -y python3 python3-pip python3-venv

# Створення робочої директорії проекту
mkdir ~/reb_system && cd ~/reb_system

# Створення та активація віртуального середовища Python
python3 -m venv venv
source venv/bin/activate

# Оновлення pip у віртуальному середовищі
pip install --upgrade pip

# Встановлення необхідних бібліотек
pip install paho-mqtt flask sqlite3 python-telegram-bot matplotlib

# sqlite3 не встановлюється через pip, бо входить до складу стандартної бі-
бліотеки Python
pip install paho-mqtt flask python-telegram-bot matplotlib

# Перевірка версій встановлених бібліотек (необов'язково, для логування)
pip freeze > requirements.txt

# Повернення до домашньої директорії (після завершення налаштування)
deactivate
cd ~
```

MQTT-брокер і клієнти

```
mosquitto_pub -h localhost -t test/topic -m "Привіт з Raspberry Pi!"

# Оновлення списку доступних системних пакетів
sudo apt update

# Встановлення самого брокера Mosquitto і утиліт для публікації та підписки
sudo apt install -y mosquitto mosquitto-clients

# Увімкнення автоматичного запуску служби брокера після кожного перезаван-
таження системи
sudo systemctl enable mosquitto
```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

```

# Ручний запуск брокера Mosquitto (виконується один раз після встановлення)
sudo systemctl start mosquitto
# Перевірка стану служби Mosquitto
# Має повертати статус: active (running)
sudo systemctl status mosquitto
# Відкриття основного конфігураційного файлу брокера Mosquitto для редагування
sudo nano /etc/mosquitto/mosquitto.conf
У файл необхідно додати наступні параметри:
port 1883
allow_anonymous true
# Перезапуск брокера Mosquitto для застосування змін конфігурації
sudo systemctl restart mosquitto
# В одному терміналі запускається підписка на тему test/topic
mosquitto_sub -h localhost -t test/topic -v
# В іншому терміналі виконується публікація повідомлення в ту саму тему
mosquitto_pub -h localhost -t test/topic -m "Привіт з Raspberry Pi!"

```

Додаток Б.3

Робота з базою даних SQLite

```

import sqlite3
conn = sqlite3.connect('/home/pi/reb_system/reb_data.db')
cur = conn.cursor()
# Основна таблиця для повідомлень/логів
cur.execute('''
CREATE TABLE IF NOT EXISTS messages (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp TEXT NOT NULL,
    topic TEXT,
    payload TEXT
)
''')
# Таблиця для збереження спектрів (values → spec_values)
cur.execute('''
CREATE TABLE IF NOT EXISTS spectrum (
    id INTEGER PRIMARY KEY AUTOINCREMENT,

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

        timestamp INTEGER NOT NULL,
        spec_values TEXT NOT NULL
    )
'''
# Таблиця для збереження аномалій
cur.execute('''
CREATE TABLE IF NOT EXISTS anomalies (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    spectrum_id INTEGER NOT NULL,
    freq INTEGER NOT NULL,
    level INTEGER NOT NULL,
    FOREIGN KEY(spectrum_id) REFERENCES spectrum(id)
)
''')
conn.commit()
conn.close()
print("Структуру бази даних створено успішно.")

```

Додаток Б.4

Налаштування логування та зчитування частоти з TinySA Ultra

```

import serial
import time
import struct
import sqlite3
from datetime import datetime

# ===== ПАРАМЕТРИ СИСТЕМИ =====
PORT = "/dev/ttyACM0" # Серійний порт підключення TinySA Ultra
BAUDRATE = 115200 # Бодрейт (швидкість обміну)
SCAN_START = 340_000_000 # Початкова частота (Гц)
SCAN_END = 2_960_000_000 # Кінцева частота (Гц)
POINTS = 2648 # Кількість точок сканування
CRITICAL_FREQS = [ # Критичні частоти для моніторингу
    433_000_000, 868_000_000, 915_000_000, 2_800_000_000
]
FREQ_TOLERANCE = 2_000_000 # Допуск від критичної частоти (Гц)
THRESHOLD = 6000 # Поріг рівня сигналу для виявлення аномалії

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

DB_PATH = "/home/pi/reb_system/reb_data.db" # Шлях до локальної БД

# ===== ЛОГ ПОВІДОМЛЕНЬ =====
def log_message(db_path, topic, payload):
    try:
        conn = sqlite3.connect(db_path)
        cur = conn.cursor()
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        payload = payload.encode('utf-8', errors='ignore').decode('utf-8',
errors='ignore')
        cur.execute(
            'INSERT INTO messages (timestamp, topic, payload) VALUES (?, ?,
?) ',
            (timestamp, topic, payload)
        )
        conn.commit()
        conn.close()
    except Exception as e:
        print("Помилка запису в messages:", e)

# ===== ЗБІР СПЕКТРУ =====
def collect_spectrum(port, baudrate, scan_start, scan_end, points):
    ser = serial.Serial(port, baudrate, timeout=10)
    time.sleep(2)
    ser.reset_input_buffer()
    scan_cmd = f"scanraw {scan_start} {scan_end} {points} unbuffered\r\n"
    ser.write(scan_cmd.encode())
    print("Команда відправлена, очікуємо відповідь...")

    data = b''
    t0 = time.time()
    while len(data) < points * 2 and time.time() - t0 < 12:
        chunk = ser.read(4096)
        if chunk:
            data += chunk
        else:
            break

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

ser.close()

print(f"Отримано {len(data)} байт.")
return data

# ===== ДЕКОДУВАННЯ СПЕКТРУ =====
def parse_spectrum(data, points):
    header_end = data.find(b'\n')
    payload = data[header_end+1:] if header_end >= 0 else data
    if payload[0:1] == b'{':
        payload = payload[1:]
    payload = payload[:points * 2]
    try:
        values = struct.unpack(f'<{points}H', payload)
    except Exception as e:
        print("Помилка розпаковки даних:", e)
        values = tuple()
    return values

# ===== ВИЯВЛЕННЯ АНОМАЛІЙ =====
def detect_anomalies(values, scan_start, scan_end, points, critical_freqs,
tolerance, threshold):
    step = (scan_end - scan_start) / points
    anomalies = []
    for idx, value in enumerate(values):
        freq = scan_start + int(idx * step)
        for c_freq in critical_freqs:
            if abs(freq - c_freq) <= tolerance and value > threshold:
                anomalies.append((freq, value))
    return anomalies

# ===== ЛОГ У БД =====
def log_to_db(db_path, timestamp, values, anomalies):
    try:
        conn = sqlite3.connect(db_path)
        cur = conn.cursor()
        cur.execute(
            'INSERT INTO spectrum (timestamp, spec_values) VALUES (?, ?)',

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

        (timestamp, ','.join(map(str, values)))
    )
    spectrum_id = cur.lastrowid
    for freq, level in anomalies:
        cur.execute(
            'INSERT INTO anomalies (spectrum_id, freq, level) VALUES
            (?, ?, ?)',
            (spectrum_id, freq, level)
        )
    conn.commit()
    conn.close()
except Exception as e:
    print("Помилка логування в БД:", e)

# ===== ГОЛОВНИЙ ЦИКЛ =====
def main():
    while True:
        raw_data = collect_spectrum(PORT, BAUDRATE, SCAN_START, SCAN_END,
POINTS)
        values = parse_spectrum(raw_data, POINTS)
        if not values:
            print("Спектр не розпізнано, пропускаємо цикл...")
            time.sleep(3)
            continue

        anomalies = detect_anomalies(values, SCAN_START, SCAN_END, POINTS,
CRITICAL_FREQS, FREQ_TOLERANCE, THRESHOLD)

        if anomalies:
            print("!!! ВИЯВЛЕНО АНОМАЛІЇ !!!")
            for freq, val in anomalies:
                print(f"Частота: {freq/1e6:.2f} МГц, Рівень: {val}")
        else:
            print("Аномалій не виявлено.")

        log_to_db(DB_PATH, int(time.time()), values, anomalies)
        time.sleep(2)

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```
# ===== СТАРТ =====
if __name__ == "__main__":
    main()
```

Додаток Б.5

Зв'язок між ЦОМ та ESP 32

```
#include <WiFi.h>
#include <PubSubClient.h>

// ==== Wi-Fi Налаштування ====
const char* ssid = "MERCUSYS_E446";
const char* password = "07031977";

// ==== MQTT Налаштування ====
const char* mqtt_server = "192.168.1.150";
const int mqtt_port = 1883;

// ==== GPIO для РЕБ ====
const int REB_PIN = 12;    // Пін для ввімкнення індикатора РЕБ

WiFiClient espClient;
PubSubClient client(espClient);

// ==== З'єднання з Wi-Fi ====
void setup_wifi() {
    delay(10);
    Serial.println("Підключення до Wi-Fi...");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nWi-Fi підключено. IP-адреса: " +
        WiFi.localIP().toString());
}

// ==== Обробка отриманих MQTT-повідомлень ====
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("[MQTT] Повідомлення в топіку: ");
    Serial.println(topic);

    // Конвертація в String
    String message;
    for (unsigned int i = 0; i < length; i++) {
        message += (char)payload[i];
    }
    Serial.println("[MQTT] Тіло: " + message);

    if (String(topic) == "reb/control") {
        if (message == "scan") {
            Serial.println("Імітація сканування...");
        }
    }
}
```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

else if (message.startsWith("activate:")) {
    int freq = message.substring(9).toInt();
    Serial.print("Команда активації РЕБ на частоті ");
    Serial.print(freq);
    Serial.println(" МГц");

    if (freq >= 450 && freq <= 2450) {
        digitalWrite(REB_PIN, HIGH);
        Serial.println("РЕБ активовано (GPIO 12 HIGH)");
        delay(5000); // Увімкнено 5 сек
        digitalWrite(REB_PIN, LOW);
        Serial.println("РЕБ вимкнено (GPIO 12 LOW)");
    } else {
        Serial.println("Частота поза допустимим діапазоном (450-2450
МГц)");
    }
}
else {
    Serial.println("Невідома команда");
}
}

// ==== Підключення до MQTT ====
void reconnect() {
    while (!client.connected()) {
        Serial.print("Підключення до MQTT-брокера...");
        if (client.connect("ESP32Client")) {
            Serial.println("Підключено");
            client.subscribe("reb/control");
            Serial.println("Підписка на топик: reb/control");
        } else {
            Serial.print("Помилка: ");
            Serial.print(client.state());
            Serial.println(" - повтор через 5 сек");
            delay(5000);
        }
    }
}

// ==== Ініціалізація ====
void setup() {
    Serial.begin(115200);
    pinMode(REB_PIN, OUTPUT); // Встановити пін як вихід
    digitalWrite(REB_PIN, LOW); // Вимкнути на старті
    setup_wifi();
    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
}

// ==== Головний цикл ====
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```
}
```

Додаток Б.6

Telegram-бот

```
from telegram import Update
from telegram.ext import ApplicationBuilder, CommandHandler, ContextTypes
import logging
import sqlite3
import os
import paho.mqtt.publish as publish

# ==== НАЛАШТУВАННЯ ====
BOT_TOKEN = "7986965404:AAE0UBgwZEgLwbTznolYT3Afj2FFFPNjpgU"
DB_PATH = "/home/pi/reb_system/reb_data.db"
MQTT_BROKER = "192.168.1.150"      # IP-адреса Raspberry Pi
MQTT_PORT = 1883
MQTT_TOPIC = "reb/control"

# ==== ЛОГУВАННЯ ====
logging.basicConfig(
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
    level=logging.INFO
)

# ==== /start ====
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Бот працює. Введіть команду. Для довідки наберіть /help")

# ==== /status ====
async def status(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Система активна. Аномалії відстежуються в реальному часі.")

# ==== /help ====
async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    help_text = (
        "Доступні команди:\n"
        "/start - привітання\n"
        "/status - поточний стан системи\n"
        "/help - список доступних команд\n"
        "/last - останні 5 аномалій\n"
        "/cmd <команда> - надіслати команду в MQTT\n"
        "/activate <частота> - активувати РЕБ в заданому діапазоні (450-2450 МГц)\n"
        "/dbsize - показати розмір бази даних"
    )
    await update.message.reply_text(help_text)

# ==== /last ====
async def last(update: Update, context: ContextTypes.DEFAULT_TYPE):
    try:
        conn = sqlite3.connect(DB_PATH)
        cur = conn.cursor()
```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

5")
    cur.execute("SELECT freq, level FROM anomalies ORDER BY id DESC LIMIT
5")
    rows = cur.fetchall()
    conn.close()

    if not rows:
        await update.message.reply_text("Аномалій не знайдено.")
        return

    response = "[Останні аномалії]\n"
    for freq, level in rows:
        response += f"Частота: {freq / 1e6:.2f} МГц – Рівень: {level}\n"

    await update.message.reply_text(response)
except Exception as e:
    await update.message.reply_text(f"Помилка доступу до БД: {e}")

# ==== /dbsize ====
async def dbsize(update: Update, context: ContextTypes.DEFAULT_TYPE):
    try:
        size = os.path.getsize(DB_PATH)
        size_kb = round(size / 1024, 2)
        await update.message.reply_text(f"Розмір бази даних: {size_kb} КБ")
    except Exception as e:
        await update.message.reply_text(f"Помилка при перевірці розміру БД:
{e}")

# ==== /cmd <текст> ====
async def cmd(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if not context.args:
        await update.message.reply_text("Введіть команду після /cmd, напри-
клад: /cmd scan")
        return

    command_text = " ".join(context.args)
    try:
        publish.single(MQTT_TOPIC, command_text, hostname=MQTT_BROKER,
port=MQTT_PORT)
        await update.message.reply_text(f"Команду '{command_text}' наді-
слано в MQTT.")
    except Exception as e:
        await update.message.reply_text(f"Помилка відправки MQTT: {e}")

# ==== /activate <частота> ====
async def activate(update: Update, context: ContextTypes.DEFAULT_TYPE):
    try:
        if len(context.args) != 1:
            await update.message.reply_text("Формат: /activate <частота в
МГц>")
            return

        freq = int(context.args[0])
        if not 100 <= freq <= 3000:
            await update.message.reply_text("Частота має бути в межах 100-
3000 МГц.")
            return

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

        payload = f"activate:{freq}"
        publish.single(MQTT_TOPIC, payload, hostname=MQTT_BROKER,
port=MQTT_PORT)
        await update.message.reply_text(f"Активация РЕБ на частоті {freq}
МГц надіслана.")
    except Exception as e:
        await update.message.reply_text(f"Помилка: {e}")

# ==== ЗАПУСК ====
if __name__ == "__main__":
    print("Запускаємо Telegram-бота...")

    app = ApplicationBuilder().token(BOT_TOKEN).build()

    app.add_handler(CommandHandler("start", start))
    app.add_handler(CommandHandler("status", status))
    app.add_handler(CommandHandler("help", help_command))
    app.add_handler(CommandHandler("last", last))
    app.add_handler(CommandHandler("cmd", cmd))
    app.add_handler(CommandHandler("dbsize", dbsize))
    app.add_handler(CommandHandler("activate", activate))

    app.run_polling()

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

requirements.txt

```

anyio==4.9.0
blinker==1.9.0
certifi==2025.6.15
click==8.2.1
contourpy==1.3.2
cyclr==0.12.1
Flask==3.1.1
fonttools==4.58.4
h11==0.16.0
httpcore==1.0.9
httpx==0.28.1
idna==3.10
itsdangerous==2.2.0
Jinja2==3.1.6
kiwisolver==1.4.8
MarkupSafe==3.0.2
matplotlib==3.10.3
numpy==2.3.1
packaging==25.0
paho-mqtt==2.1.0
pillow==11.2.1
pyparsing==3.2.3
python-dateutil==2.9.0.post0
python-telegram-bot==22.1
six==1.17.0
sniffio==1.3.1
typing_extensions==4.14.0

```

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

Werkzeug==3.1.3

		Скиданчук В.Д.			ККІКБ.КР.Б-123-25-ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		89