НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Радіотехнічний факультет

Кафедра радіотехнічних систем

До захисту допущено:

Завідувач кафедри

_____ Сергій ЖУК

«7» червня 2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Радіотехнічні комп'ютеризовані системи»

спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Система передачі координат роботизованої платформи з використанням технології LoRa»

Виконав (-ла):
студент (-ка) IV курсу, групи РС-01
Новоженя Володимир Андрійович _____

Керівник:
к.т.н., доцент
Могильний Сергій Борисович _____
_____

Рецензент:
к.т.н., доцент кафедри РІ
Літвінцев Сергій Миколайович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Радіотехнічний факультет
Кафедра радіотехнічних систем

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – **172 «Телекомунікації та радіотехніка»**
Освітня програма «**Радіотехнічні комп'ютеризовані системи**»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій ЖУК
«16» квітня 2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Новожені Володимиру Андрійовичу

1. Тема роботи «Система передачі координат роботизованої платформи з використанням технології LoRa»,

керівник роботи Могильний Сергій Борисович, доцент, к.т.н.,
затверджені наказом по університету від «29» травня 2024 р. №2178

2. Термін подання студентом роботи 13.06.2024 р.

3. Вихідні дані до роботи Розробити систему передачі координат роботизованої платформи на основі технології LoRa. Провести дослідження реальної дальності зв'язку системи в умовах міста. _____

4. Зміст роботи 1. Вступ. 2. Розробка концепції системи. 3. Огляд ринку мікрокомп'ютерів та модулів LoRa та GNSS, вибір складових системи. 4. Збірка експериментального макету системи. 5. Розробка програмного забезпечення, налаштування системи. 6. Опис та результати експериментального дослідження. _____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) 1. Структурна схема експериментального макету. 2. Блок-схема алгоритму роботи системи. 3. Графік потужності сигналу роботизованої платформи на шлюзі в залежності від відстані. _____

6. Дата видачі завдання 16 квітня 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд та аналіз технічної літератури	до 22.04.2024 р.	
2.	Розробка концепції та структурної схеми системи	до 29.04.2024 р.	
3.	Збірка експериментального макету системи	до 06.05.2024 р.	
4.	Налаштування системи, тестування в лабораторії	до 20.05.2024 р.	
5.	Проведення експерименту	до 27.05.2024 р.	
6.	Написання пояснювальної записки	до 12.06.2024 р.	
7.	Підготовка презентації	до 16.06.2024 р.	

Студент Володимир, НОВОЖЕНЯ

Керівник Сергій, МОГИЛЬНИЙ

АНОТАЦІЯ

Обсяг пояснювальної записки дипломної роботи становить 68 сторінок, які включають в себе 5 розділів, 28 ілюстрацій, 2 таблиці, 3 додатки і 4 бібліографічне найменування за переліком джерел посилань.

Актуальність теми: Використання роботизованих платформ для виконання завдань в умовах, коли є небезпека для життя людини, вимагає від радіоканалу, через який здійснюється зв'язок між оператором та платформою (в тому числі і передача даних про місцезнаходження платформи) достатньої дальності зв'язку та захищеності. При цьому відстань між оператором і платформою може досягати 5-10 км, що не можна реалізувати з технологією Wi-Fi. А мобільний зв'язок в умовах роботи платформи не лише часто відсутній, а навіть заборонений для використання. Застосування технології LoRa може забезпечити достатню дальність зв'язку між платформою та оператором, а також надійність, стійкість до завад, захищеність радіоканалу від несанкціонованого доступу.

Мета дослідження: Розробити надійну систему передачі даних про місцезнаходження роботизованої платформи за допомогою технології LoRa. Провести дослідження реальної дальності зв'язку між передавачем та приймачем LoRa в умовах міста.

Об'єкт дослідження: передача координат за допомогою технології LoRa.

Предмет дослідження: побудова системи передачі координат роботизованої платформи.

Новизна одержаних результатів: розроблена та налаштована проста в виготовленні система, що дозволить підвищити безпеку та дальність передачі координат від платформи до оператора. Визначена дальність дії системи в умовах міської забудови.

Ключові слова: РОБОТИЗОВАНА ПЛАТФОРМА, ШЛЮЗ, ВЕБ-СЕРВЕР, GPS, GNNS, LORA, RASPBERRY PI, FLASK, SQLITE.

ANOTATION

The volume of the explanatory note for the diploma thesis is 68 pages, which includes 5 chapters, 28 illustrations, 2 tables, 3 appendices, and 4 bibliographic entries in the list of references.

Relevance of the topic: The use of robotic platforms for performing tasks in conditions where there is a danger to human life requires the radio channel, through which communication between the operator and the platform is carried out (including the transmission of data about the platform's location), to have sufficient communication range and security. The distance between the operator and the platform can reach 5-10 km, which cannot be achieved with Wi-Fi technology. Moreover, mobile communication in the conditions of the platform's operation is not only often unavailable but even prohibited for use. The application of LoRa technology can ensure sufficient communication range between the platform and the operator, as well as reliability, resistance to interference, and protection of the radio channel from unauthorized access.

Objective of the research: To develop a reliable data transmission system for the location of the robotic platform using LoRa technology. To study the actual communication range between the LoRa transmitter and receiver in urban conditions.

Object of the research: transmission of coordinates using LoRa technology.

Subject of the research: construction of a coordinate transmission system for the robotic platform.

Novelty of the obtained results: a simple-to-manufacture system has been developed and configured, which will increase the safety and range of coordinate transmission from the platform to the operator. The communication range of the system in urban conditions has been determined.

Keywords: ROBOTIC PLATFORM, GATEWAY, WEB SERVER, GPS, GNNS, LORA, RASPBERRY PI, FLASK, SQLITE.

Пояснювальна записка
до дипломної роботи
на тему: «Система передачі координат роботизованої платформи з використанням технології LoRa»

Київ – 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – програмне забезпечення

ОС – операційна система

IoT – інтернет речей (Internet of Things)

GNSS – global navigation satellite system (супутникова система навігації)

GPS – global positioning system (система глобального позиціонування)

LIDAR – Light Identification, Detection and Ranging (Світлова ідентифікація, виявлення та визначення дальності)

RSSI – Received Signal Strength Indication (індикатор потужності прийнятого сигналу)

ЛЧМ – лінійна частотна модуляція

ВСТУП

В сучасному світі все більше обертів набирає використання різноманітних роботизованих платформ, починаючи від роботів, інтегрованих в IoT систему міста для доставки різноманітних товарів, і закінчуючи транспортними, евакуаційними та повноцінними бойовими військовими платформами для виконання завдань в небезпечних для життя людей умовах. До того ж, активно розвиваються системи з машинним зором, які можуть дозволити роботизованим платформам аналізувати простір довкола них через встановлені відеокамери і самостійно вибудовувати маршрут до пункту призначення, оминаючи перешкоди. Роботизовані платформи стають все більш автономними та незалежними від оператора.

Проте, актуальним залишається питання визначення місцеположення платформи. Знання цієї інформації необхідне як для побудови маршруту поїздки платформи, так і для моніторингу її місцеположення оператором, на випадок, якщо платформа десь застрягне або вийде з ладу, і її можна було знайти. Тоді постає питання, яким чином можна надійно передавати координати платформи так, щоб це не потребувало значних затрат електроенергії (оскільки заряд акумулятора платформи обмежений), займало небагато місця, було відносно дешевим та простим у використанні і, насамперед, забезпечувало достатню дальність покриття та надійність каналу зв’язку.

Із дешевих та доступних варіантів можна розглянути декілька. Технологія Wi-Fi хоч і забезпечує швидку передачу великих об’ємів даних з достатнім рівнем захисту, проте не може бути застосована у випадках, коли відстань між платформою та оператором може сягати до 10 км. Мобільний зв’язок хоч має більшу зону покриття ніж Wi-Fi, проте, в умовах роботи роботизованої платформи він може бути не лише часто відсутній, а навіть заборонений для використання з міркувань безпеки. Ще одним перспективним варіантом є технологія LoRa.

1 РОЗРОБКА КОНЦЕПЦІЇ СИСТЕМИ

Технологія, завдяки якій буде здійснюватися передача даних про місцерозташування роботизованої платформи на сервер, – це технологія LoRa.

LoRa – технологія та однойменний метод модуляції, запатентовані компанією Semtech. Метод модуляції LoRa є різновидом ЛЧМ з технологією розширення спектра, при якій дані кодуються в широкосмугові імпульси (рис. 1.1) з частотою, що збільшується чи зменшується на певному часовому інтервалі часу (рис. 1.2). Таким чином, на відміну від технології прямого розширення спектра, приймач LoRa є стійким до відхилення частоти від номінального значення. Працює в дециметровому діапазоні у вільних від ліцензування діапазонах (433 МГц в Україні).

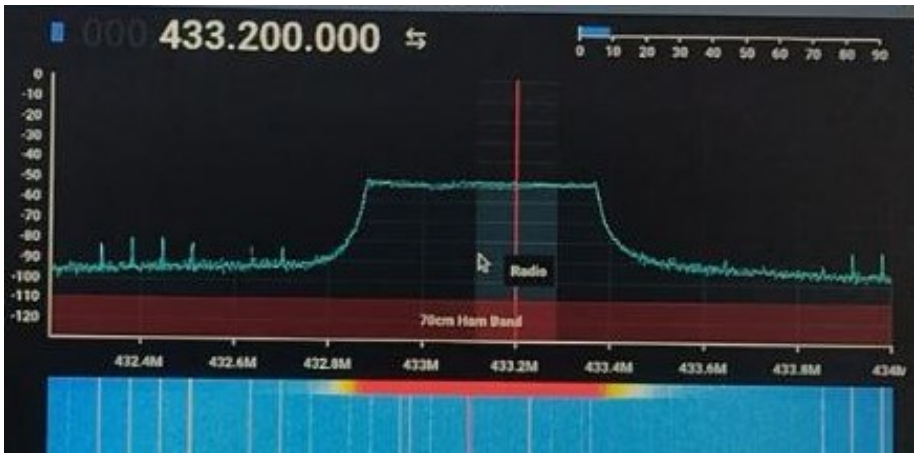

Рисунок 1.1 – Широкосмуговий сигнал LoRa зафіксований SDR приймачем

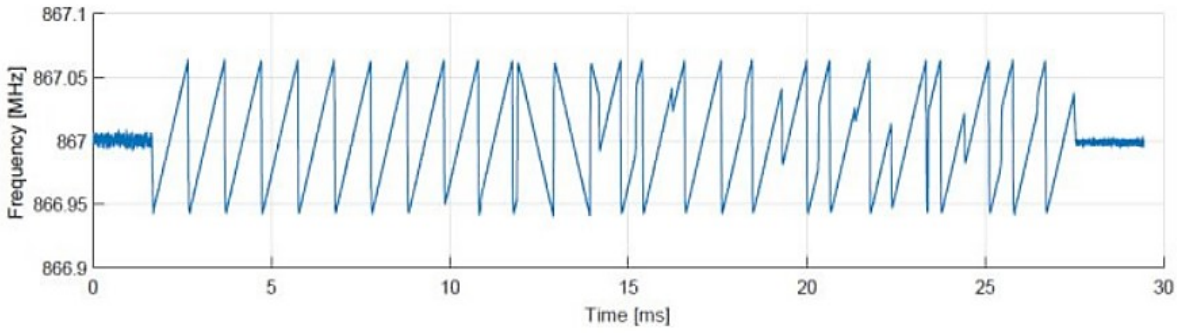

Рисунок 1.2 – Демонстрація зміни частоти сигналу LoRa з часом

LoRa дозволяє демодулювати сигнали на рівні 20 дБ нижче рівня шумів, тоді як більшість систем з частотною маніпуляцією можуть коректно працювати з сигналами на рівні не ниже 10 дБ над рівнем шумів. Завдяки цьому, а також чутливості приймача на рівні -150 дБм, технологія LoRa здатна забезпечувати міжмашинний зв'зок з низьким споживанням електроенергії на відстанях до 15 км на пересічній місцевості, та до 5 км в умовах щільної міської забудови.

Швидкість передачі даних може змінюватися від 300 біт/с до 50 Кб/с, при цьому швидкість передачі пропорційно зменшується в залежності від відстані між передавачем та приймачем.

Серед характерних переваг технології LoRa можна також виділити:

- гнучке налаштування смуги пропускання;
- стійкість до впливу завад;
- стійкість до зміщення частоти через ефект Допплера;
- широка пропускна здатність та підтримка зв'язку з до 1000000 пристроїв (у рамках протоколу LoRaWAN).

Безпека радіоканалу забезпечується за рахунок підтримки дворівневого шифрування. Кожен успішно зареєстрований в мережі пристрій отримує AES128 ключ на одну сесію, який шифрує повідомлення на канальному рівні, а системне повідомлення кодується власним AES128 ключем, що гарантує конфіденційність передачі інформації від пристрою до шлюзу і навпаки.

Для отримання інформації про місцеположення платформи було вирішено використати підсистему, здатну працювати з GNSS, тобто використовувати не тільки одну з доступних систем супутникового зв'язку (такі як GPS чи Galileo), а одразу мати змогу визначати координати завдяки усім доступним супутникам навігації та наземним радіомаякам, що значно підвищить як точність отримуваних координат, так і надійність системи.

За основу системи вирішено обрати мікрокомп’ютери від компанії Raspberry Pi Foundation. Мікрокомп’ютери даного виробника відрізняються своєю доступність та низькою вартістю в порівнянні зі своїми аналогами. Raspberry Pi дуже гнучка платформа, здатна працювати на різних операційних системах, має підтримку великої кількості периферійних пристроїв за рахунок наявності різноманітних портів та інтерфейсів (таких як GPIO, I2C, UART), через що користується популярністю в навчальних програмах та системах IoT. Raspberry Pi Foundation надають якісну та детальну документацію на свою мікрокомп’ютери. Наявність активної та великої спільноти користувачів та розробників спрощує та пришвидшує роботу над проектами завдяки великій кількості готових, доступних та робочих рішень.

Потужності Raspberry Pi забагато для вирішення питання керування двома модулями. Проте, роботизована платформа, на якій встановлена система передачі координат, в майбутньому може бути використана для інтеграції в неї значно більшої кількості модулів та систем, таких як LIDAR та система з машинним зором, де запас обчислювальної потужності Raspberry Pi знадобиться.

В якості роботизованої платформи обрано PiRacer AI Kit від Waveshare[1].

Система передачі координат складається з двох пристроїв – роботизована платформа і сервер.

Роботизована платформа має вбудований блок живлення, за рахунок якого і живиться мікрокомп’ютер з ОС Raspbian. До портів Raspberry Pi підключені модуль GNSS та передатчик LoRa.

Сервер складається з ідентичного мікрокомп’ютера Raspberry Pi, аби уникнути проблем з інтерпретацією повідомлень, та встановленого на нього модуля LoRa, що виконує роль шлюзу. Сервер стаціонарний, встановлений в 505 лабораторії на 5 поверсі Радіотехнічного факультету в 17 корпусі.

Система працює наступним чином: після увімкнення роботизованої платформи на ній в ручну запускається скрипт, який кожні декілька секунд зчитує дані з модуля GNSS, трансформує їх в координати формату «широта, довгота» і відправляє повідомлення на шлюз сервера. На увімкнутому сервері, в цей час, запущений скрипт, що відслідковує появу сигналу від платформи на шлюзі, і при його появі видобуває з повідомлення координати і записує їх в базу даних SQLite разом зі значенням потужності прийнятого сигналу. Окремо виконується скрипт, що запускає веб-сторінку. На ній будуть відображатися дата і час на сервері, останні отримані координати платформи та мапа з точками, які відображають потужність прийнятого на вході шлюзу сигналу та місцеположення платформи під час передачі сигналу. Дана мапа дозволить не тільки відслідковувати місцеположення платформи, а й дасть змогу дослідити дальність зв’язку між приймачем та шлюзом LoRa даної системи, оцінити затухання сигналу з відстанню та проаналізувати як різноманітні перешкоди впливають на дальність зв’язку.

В ході експериментальної частини роботи буде проведено збір даних, для аналізу дальності зв’язку системи. Оскільки в рамках роботи буде проведене лише налаштування системи передачі координат, роботизована платформа переміщуватиметься не своїм ходом.

2 ОГЛЯД РИНКУ МІКРОКОМП'ЮТЕРІВ ТА МОДУЛІВ LORA ТА GNSS, ВИБІР СКЛАДОВИХ СИСТЕМИ

2.1 Вибір мікрокомп'ютера

Лінійка мікрокомп'ютерів від Raspberry Pi Foundation налічує в собі багато моделей різних років випуску, обчислюваних потужностей та форм-факторів. На розгляд були обрані найновіші моделі Raspberry: 3B+, 4 і Raspberry Pi 5. Для наочного порівняння ключових для розробки системи характеристик мікрокомп'ютерів, порівняння моделей Raspberry Pi наведене у вигляді табл. 2.1.1.

Таблиця 2.1.1 – Порівняння ключових характеристик різних моделей Raspberry Pi:

	Raspberry Pi 3 B+	Raspberry Pi 4	Raspberry Pi 5
Процесор	64-бітний, 4-ядерний		
Оперативна пам'ять	4 ГБ		
Тактова частота	1,4 ГГц	1,5 ГГц	2,4 ГГц
Рекомендований струм PCU[2]	2,5 А	3,0 А	5,0 А
Середній споживаний струм[2]	500 мА	600 мА	800 мА

Raspberry Pi 5 відрізняється від попередніх моделей вищою продуктивність та швидкодією. Потужніший процесор, вищі частоти роботи процесора, порти з більшою пропускною здатністю, досконаліший чіпсет, можливість використовувати більш досконале та функціональне ПЗ. Проте,

платою за це є високе енергоспоживання, що є проблемою коли мова йде про використання роботизованих платформ, де окрім мікрокомп’ютера серед потужних споживачів заряду акумулятора наявні електродвигуни.

Raspberry Pi 3 B+ має значно нижче енергоспоживання, але й має менше можливостей. Так як розроблювана система передачі координат для роботизованої платформи у майбутньому може бути доповнена більш потужними інструментами, такими як машинний зір або LIDAR, для яких обчислювальних потужностей даної моделі мікрокомп’ютера може не вистачити.

Raspberry Pi 4 (рис. 2.1.1) в свою чергу є оптимальним варіантом для розроблюваної системи. Він має дещо кращі обчислювальні потужності аніж 3 модель, і в тож же час споживає значно менше електроенергії, на відміну від 5 распбері. Тому, розроблювана в рамках даної дипломної роботи система побудована на базі мікрокомп’ютера Raspberry Pi 4.

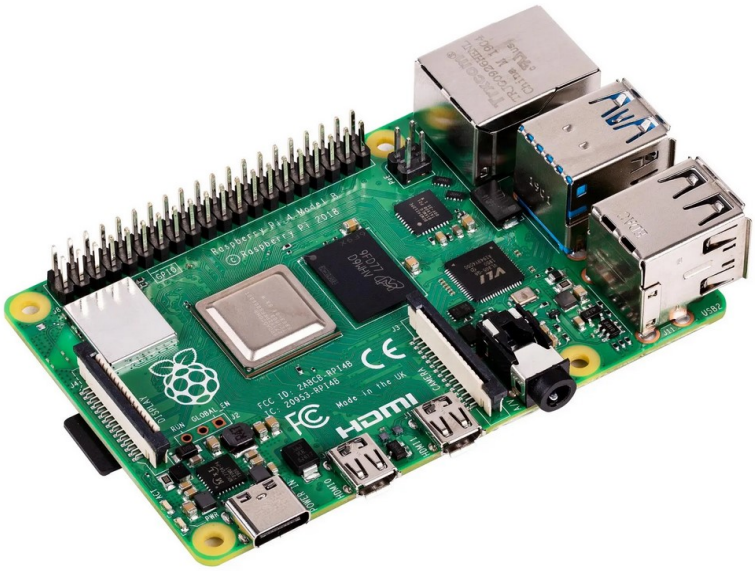

Рисунок 2.1.1 – Мікрокомп’ютер Raspberry Pi 4 4GB

2.2 Вибір модуля GNSS

Серед доступних на ринку модулів GNSS було вирішено використати модуль L76X Multi-GNSS HAT від Waveshare (рис. 2.2.1), оскільки даний модуль розроблений для роботи з Raspberry Pi, має роз'єми GPIO та Micro USB, взаємодіє з іншими пристроями через інтерфейс UART, відслідковує до 33 каналів, що забезпечує високу точність позиціонування, та працює з супутниками систем GPS, BD2 та QZSS.

Рисунок 2.2.1 – Модуль L76X Multi-GNSS HAT від Waveshare з антеною

Деякі з важливих характеристик модуля:

– споживання струму – 13 мА;

– робоча напруга – 5 В;

– робочі частоти – GPS – 1575,42 МГц, BD2 – 1561,098 МГц;

– чутливість приймача при відслідковуванні – -163 дБм;

– точність позиціонування – до 2,5 м;

Більш детальну інформацію про модуль можна переглянути на сайті виробника [3].

2.3 Вибір модуля LoRa

Серед розглянутих варіантів готових модулів LoRa на сайтах українських торгових мереж радіокомпонентами (таких як РКС Компоненти та ARDUINO.UA) було вирішено обрати модуль SX1268 433M LoRa HAT від Waveshare (рис. 2.3.1).

Рис. 2.3.1 – Модуль SX1268 433M LoRa HAT від Waveshare

Даний модуль працює на частоті 433 МГц, виконаний на основі мікросхеми SX1268 від Semtech і спеціально розроблений для роботи з Raspberry Pi. На платі встановлені роз’єми GPIO та Micro USB. Взаємодія з модулем відбувається через інтерфейс UART. Заявлені виробником характеристики модуля наступні:

– споживання струму під час передачі сигналу – 100 мА;

– споживання струму під час отримання – 11 мА;

– споживання струму в сплячому режимі – 2 мкА;

– максимальна потужність випромінювання – 22 дБм;

– максимальна довжина повідомлення – 240 байт;

– буфер – 1000 байт;

– робочий діапазон частот – від 410,125 МГц до 493,125 МГц;

– мінімальна швидкість передачі – 0,3 Кб/с;

– максимальна швидкість передачі – 62,5 Кб/с;

– робоча напруга – 5 В;

– чутливість приймача (за швидкості передачі 0,3 Кб/с) – -147 дБм;

– потенційна дальність – 5 км.

Даний модуль підтримує приватний протокол передачі та не підтримує LoRaWAN. Підтримує передачу по мережі типу «точка-точка», широкомовну передачу, моніторинг каналів і багаторівневу ретрансляцію для наддалекого зв’язку. Модуль прослуховує канал перед тим, як надіслати ним повідомлення, аби упевнитися що канал чистий, що в свою чергу підвищує надійність передачі. Наявний індикатор сигналу RSSI для оцінки якості сигналу, підтримка пробудження по радіо, онлайн-конфігурація, визначення несучої, автоматичне реле, ключ зв'язку, та режим сну з низьким енергоспоживанням. Більш детальну інформацію про модуль можна переглянути на сайті виробника [4].

Реальні чутливість приймача системи, потенційна дальність, а також здатність модуля розрізняти сигнали на 20 дБ нижче рівня шумів є метою експериментального дослідження.

3 ЗБІРКА ЕКСПЕРИМЕНТАЛЬНОГО МАКЕТУ СИСТЕМИ

Згідно розробленої концепції системи в 1-му розділі та обраних модулів в 2-му розділі, було розроблено структурну схему системи передачі координат роботизованої платформи за допомогою технології LoRa у вигляді блок-схеми на рис. 3.1.

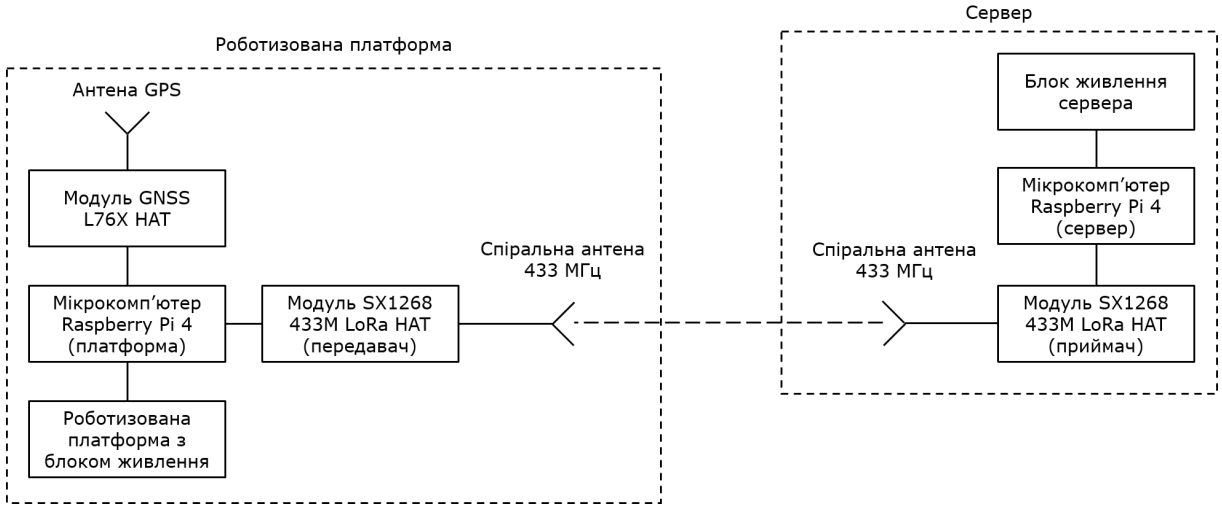

Рисунок 3.1 – Структурна схема системи

Система складається з двох підсистем (роботизована платформа та сервер), між якими встановлено зв'язок через радіоканал LoRa. Детальніші параметри радіоканалу будуть визначені під час налаштування системи. Загалом, система включає в себе: два мікрокомп'ютери Raspberry Pi 4 на 4 ГБ оперативної пам'яті, два модулі SX1268 433M LoRa HAT, дві спіральні антени під діапазон 433 МГц, один модуль L76X GNSS HAT, одна антена модуля GNSS, роботизована платформа з вбудованими акумуляторами та один блок живлення на 5 вольт для сервера.

3.1 Збірка роботизованої платформи

На встановлений на роботизованій платформі мікрокомп'ютер Raspberry Pi 4 на роз'єми GPIO додається модуль LoRa таким чином, щоб роз'єми GPIO на Raspberry та LoRa відповідали один одному. До роз'ємів GPIO на модулі LoRa підключається дроти, що живлять систему від акумуляторів платформи (слідкуйте за відповідністю роз'ємів GPIO на роботизованій платформі та LoRa). На коаксіальний з'єднувач встановлено спіральну антену на 433 МГц.

Модуль GNSS встановлюється на нейлонові стійки і через роз'єм Micro USB підключається до одного з доступних портів USB мікрокомп'ютера. Антена GNSS через адаптер підключається до коаксіального порту на платі модуля. На зворотній стороні плати модуля встановлюємо батарейку ML1220.

В слот micro SD встановлюється картка пам'яті на 32 ГБ з попередньо завантаженою на неї ОС Raspbian та активованим SSH та VNC. Для налаштування ПЗ, роботизована платформа підключається до мережі інтернет через Ethernet порт.

На рис. 3.1.1 зображено зібраний експериментальний макет роботизованої платформи.

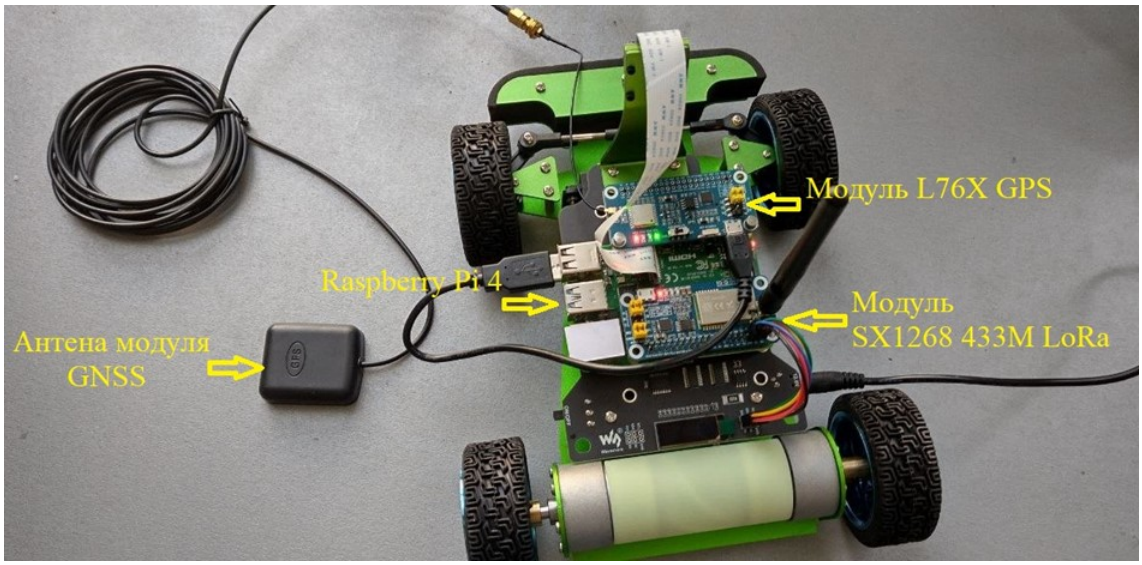

Рисунок 3.1.1 – Макет роботизованої платформи з позначеннями модулів

Для налаштування роботи модулів в необхідному режимі, необхідно встановити перемички у відповідні положення. На модулі GNSS перемички мають стояти на позиції ‘A’. На модулі LoRa перемички мають стояти на позиції ‘B’, а також має стояти перемичка між M0 та GND. На рис. 3.1.2 добре видно як саме підключаються кабелі та перемички.

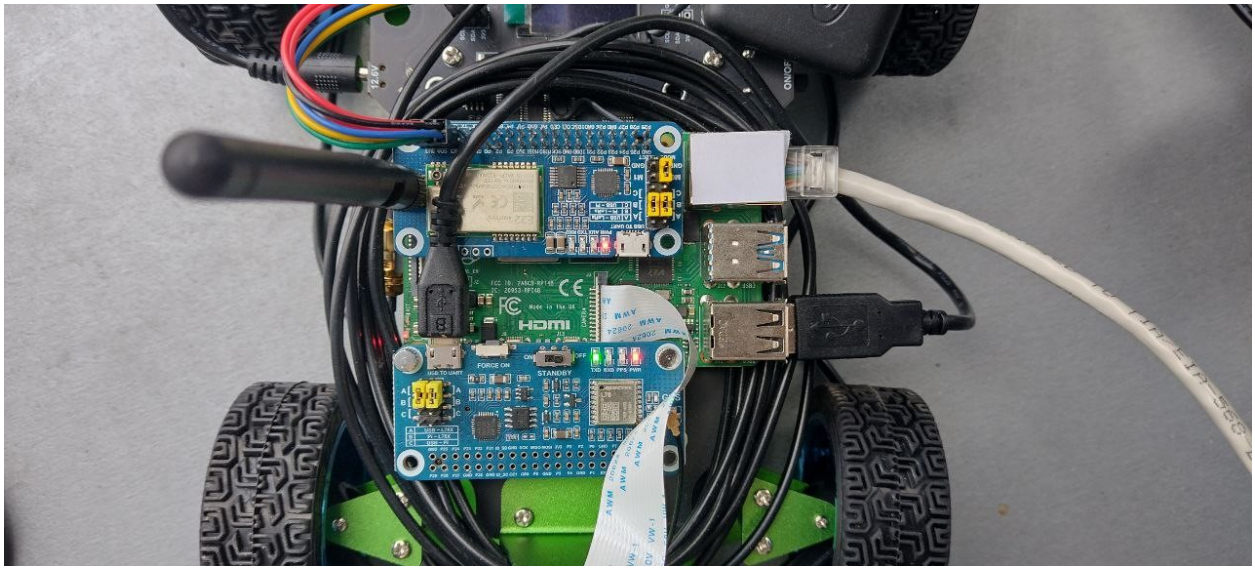

Рисунок 3.1.2 – Роботизована платформа, підключення кабелів та перемичок

Перед першим увімкнення варто повністю зарядити акумулятор. Під час налаштування платформи слід користуватися живленням від мережі.

3.2 Збірка сервера

На мікрокомп’ютер Raspberry Pi 4, що виконує роль сервера, на роз’єми GPIO встановлюється модуль LoRa таким чином, щоб роз’єми GPIO на Raspberry та LoRa відповідали один одному. На коаксіальний з’єднувач на модулі LoRa встановлено спіральну антену на 433 МГц.

Для налаштування роботи модуля LoRa в необхідному режимі, перемички мають стояти на позиції ‘B’, а також має стояти перемичка між M0 та GND.

В слот micro SD встановлюється картка пам’яті на 32 ГБ з попередньо завантаженою на неї ОС Raspbian та активованим SSH та VNC. Для налаштування ПЗ, роботизована платформа підключається до мережі інтернет через Ethernet порт. Живиться сервер від мережевого адаптера 5 В.

На рис. 3.2.1 зображено зібраний та змонтований на стенді в 505 лабораторії в корпусі Радіотехнічного факультету експериментальний макет сервера системи з встановленим на нього шлюзом LoRa.

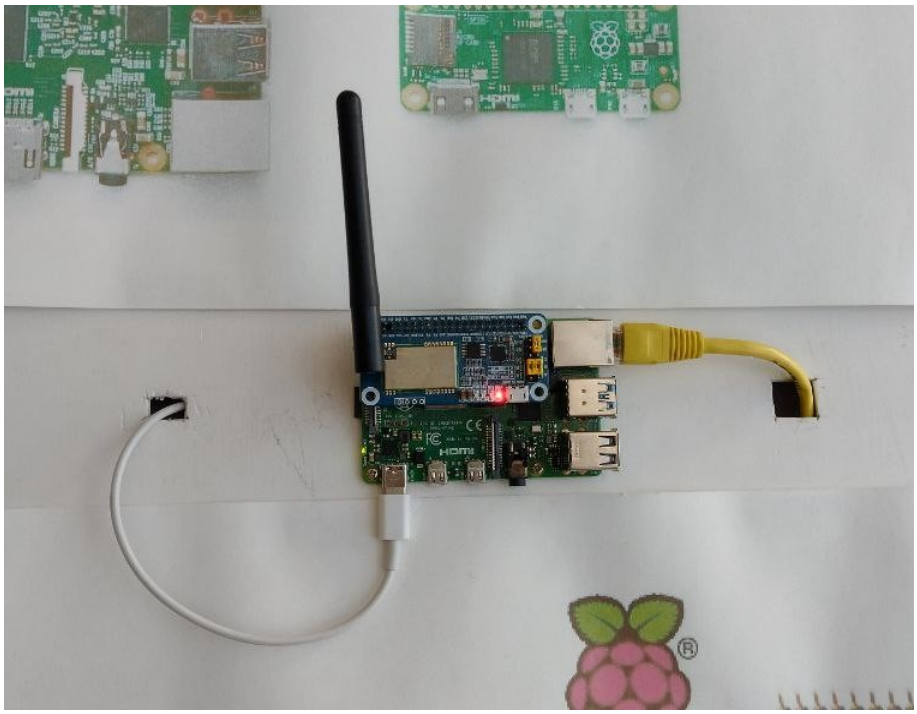

Рисунок 3.2.1 – Сервер, встановлений в 505 лабораторії на 5 поверсі в 17 корпусі, радіотехнічний факультет

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, НАЛАШТУВАННЯ СИСТЕМИ

4.1 Алгоритм роботи системи

Згідно концепції системи було розроблено алгоритм, за яким працюватиме розроблювана системи і на основі якого буде створено ПЗ системи на мові програмування Python. Сам алгоритм наведений на рис. 4.1.1 у вигляді блок-схеми.

Рисунок 4.1.1 – Блок-схема алгоритму роботи системи передачі координат

4.2 Налаштування платформи

Для підключення до Raspberry Pi і налаштування системи скористаємося сервісом VNC Viewer, з попередньо встановленим на ПК додатком RealVNC Viewer для роботи з сервісом (рис. 4.2.1). При використаному методі підключення до мікрокомп'ютера необхідно щоб він та ПК з якого відбувається підключення були в одній мережі.

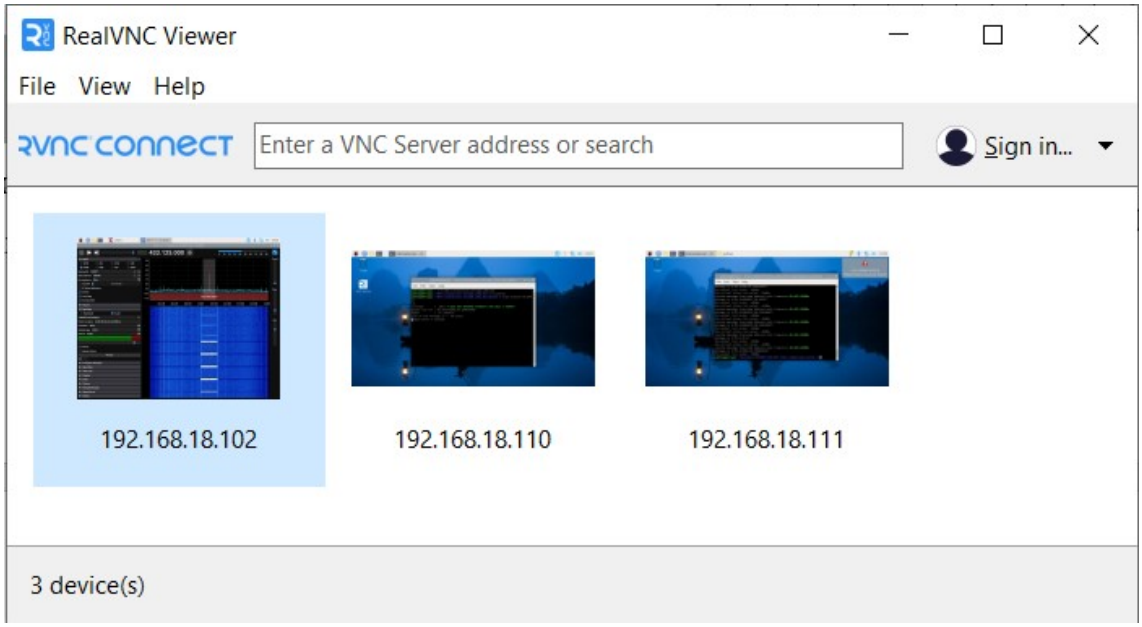

Рисунок 4.2.1 – Інтерфейс RealVNC Viewer

Для підключення до распбері необхідно знати його ip-адресу. Для цього можемо скористати програмою або веб-срвісом IP Scanner (рис. 4.2.2), яка сканує мережу на наявність підключених пристроїв.

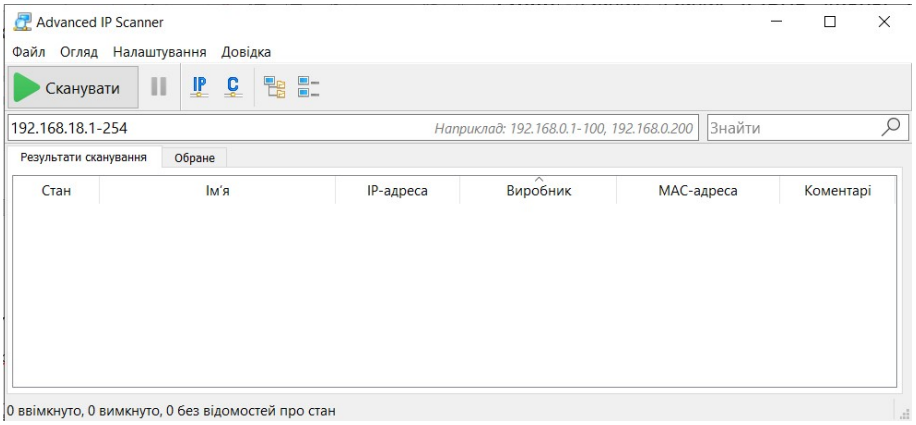

Рисунок 4.2.2 – Вікно програми IP Scanner

Знаючи MAC-адресу порта Ethernet нашого распбері, або якщо до мережі підключений тільки один распбері, ми леко дізнаємося необхідну нам ip-адресу.

Вводимо в пошукове вікно RealVNC ip-адресу мікрокомп'ютера, що встановлений на роботизовану платформу, у вікні авторизації вводимо логін та пароль користувача і все, ми підключилися до распбері.

Відкриємо в головному меню вкладку «Preferences», «Raspberry Pi Configuaration» (рис. 4.2.3), «Interfaces» та активуємо усі інтерфейси (рис. 4.2.4).

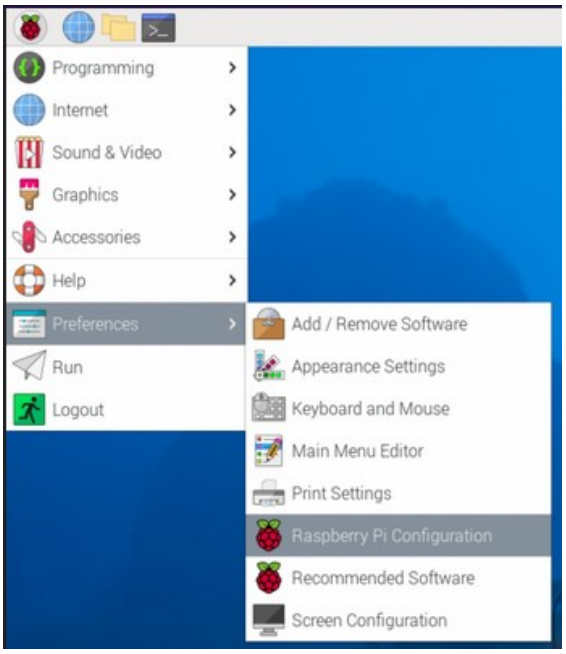

Рисунок 4.2.3 – Шлях до вкладки Raspberry Pi Configuration

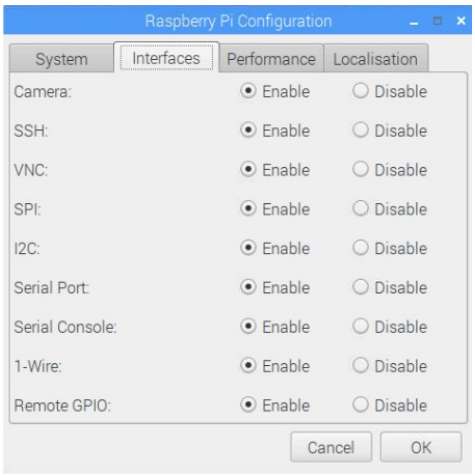

Рисунок 4.2.4 – Активація інтерфейсів на Raspberry Pi

Тепер, налаштуємо порт Serial. Відкриваємо на панелі меню Command shell і вводимо команду «`sudo raspi-config`», обираємо «Interfacing Options», «Serial», «No», «Yes» (рис. 4.2.5).

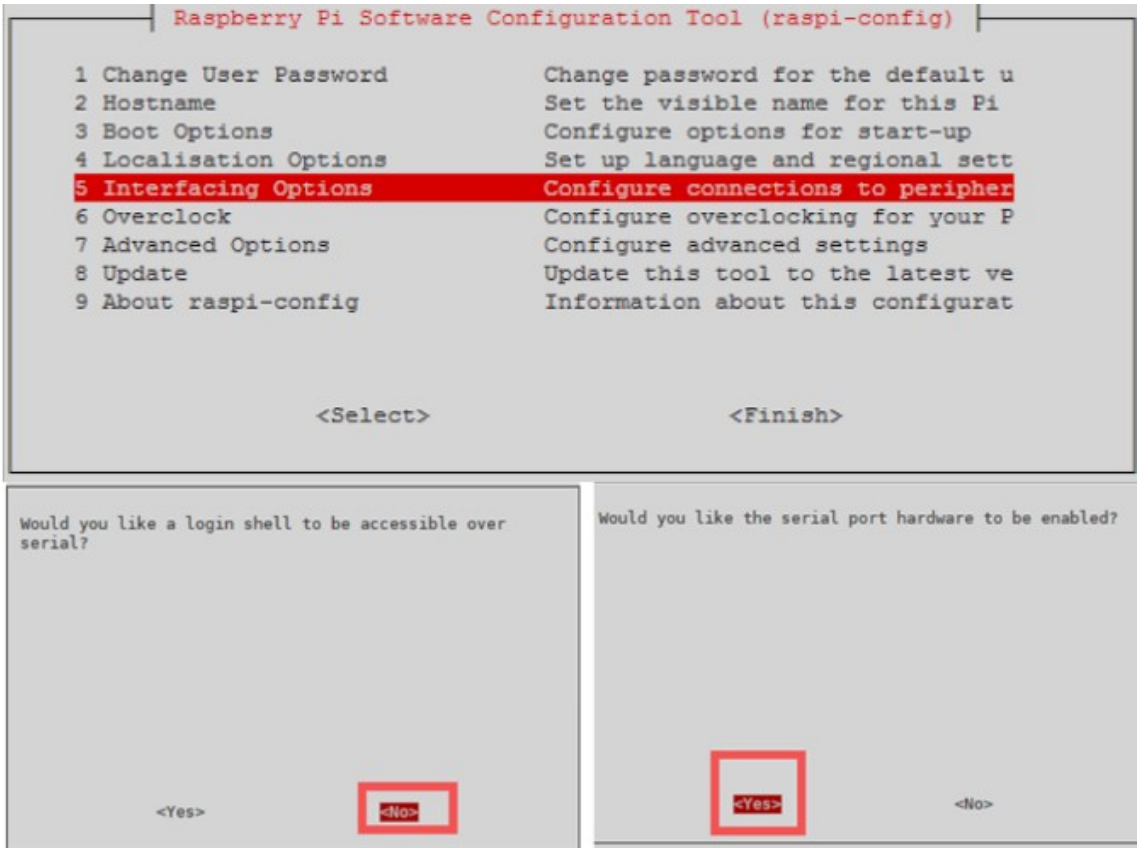

Рисунок 4.2.5 – Налаштування порту Serial

Після цього, по черзі вводимо наступні команди:

```
sudo apt update
sudo apt upgrade
sudo apt-get install python3
sudo apt-get install python3-pip
sudo pip3 install RPi.GPIO
sudo apt-get install python-serial
sudo reboot
```

Таким чином, оновлюємо ОС Raspberry, бібліотеку мови програмування Python, інсталлер доповнень pip, бібліотеки для роботи з GPIO та Serial, після чого система перезапускається для завершення оновлення. Підключаємося до Raspberry знову і продовжуємо працювати над створенням ПЗ системи.

4.2.1 Налаштування отримання координат платформи

Виробник модуля GNSS на своєму сайті пропонує нам готове програмне рішення, що видобуває дані про місцезнаходження платформи з модуля, трансформує їх у координати для різних систем навігації. Його використання дозволить значно спростити процес розробки, так як відпадає необхідність детально вивчати машинні коди для роботи з чіпом модуля GNSS. Дане рішення чудово підійде для даної роботи.

Перш за все, встановимо необхідні для роботи GNSS бібліотеки gpsd, pynmeagps та gps3. Введемо в Command shell та виконаємо наступні команди:

```
sudo apt-get install gpsd gpsd-clients python-gps
sudo pip3 install pynmeagps
sudo pip3 install gps3
```

Завантажимо демо-версію коду для роботи з модулем:

```
Wget
https://files.waveshare.com/upload/f/f4/L76X_GPS_HAT_Code.zip
unzip L76X_GPS_HAT_Code.zip
sudo chmod 777 -R L76X_GPS_HAT_Code
```

Так як модуль GNSS підключений до распбері через порт USB, треба дізнатися до якого саме порту програмі треба звертатися аби отримати доступ до модуля. Виконаємо команду `dmesg | grep tty` щоб дізнатися назви задіяних портів (рис. 4.2.1.1).

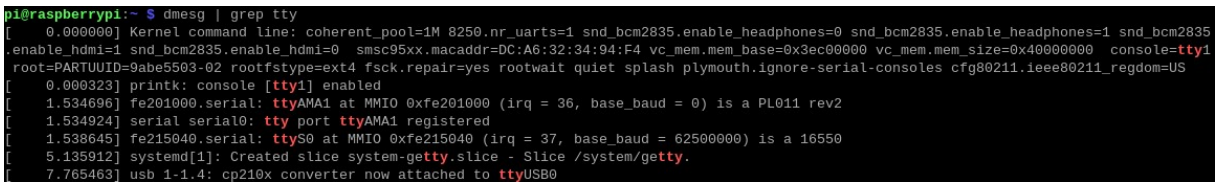

Рисунок 4.2.1.1 – Визначення задіяних портів Raspberry Pi

Як бачимо на рис 4.2.1.1, до порту `ttyUSB0` підключений UART конвертер cp210x – мікросхема встановлена на модулі GNSS що забезпечує зв’язок між мікросхемою GNSS та распбері.

На робочому столі Raspberry запускаємо Провідник, відкриваємо теку з файлами програми за адресом L76X_GPS_HAT_Code/RaspberryPi/python та за допомогою Thonny відкриваємо файл config.py і замінюємо в його коді всі ttyS0 на ttyUSB0. Таким чином, при запуску програма звертатиметься на порт до якого підключено модуль GNSS.

Тепер можемо запустити демо версію програми та перевірити роботу модуля. В Command shell по черзі вводимо та виконуємо наступні команди:

```
cd L76X_GPS_HAT_Code/RaspberryPi
cd python
sudo python3 main.py
```

Під час роботи програми, антена модуля GNSS має бути біля вікна, або краще надворі, щоб модуль міг побачити якнайбільше супутників. Якщо все зроблено правильно, після 30 секунд під час першого запуску матимемо схожий результат як на рисунку 4.2.1.2. У вікні Command shell відображатимуться висота платформи відносно рівня моря, координати у декількох форматах для 4 різних систем – wgs84, Google Map, Amap, Baidu, та швидкість переміщення платформи, кожні 10 секунд координати будуть оновлюватися.

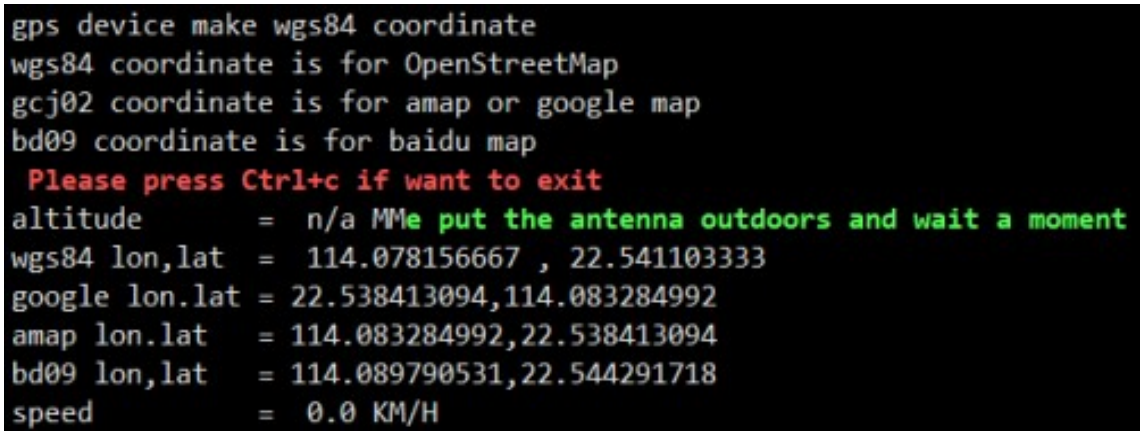

Рисунок 4.2.1.2 – Приклад роботи демо-версії коду для модуля GNSS

Таким чином, ми налаштували модуль GNSS і вже можемо отримувати інформацію про місцезнаходження платформи.

Тепер, за допомогою команди cd вийдемо в кореневий каталог і створимо папку для робочих файлів системи:

```
cd
mkdir GPS_transmission_through_LoRa_Hat
cd GPS_transmission_through_LoRa_Hat
mkdir python
cd
```

Відкриємо папку L76X_GPS_HAT_Code/RaspberryPi/python в провіднику та скопіюємо весь її вміст (включаючи файли і папки) в папку GPS_transmission_through_LoRa_Hat/python. Перейменуємо файл main.py в TR_GPS.py та відкриємо його за допомогою Thonny. Видаляємо з коду все що стосується відображення координат не за стандартом Google Map для оптимізації роботи програми (див. Додаток А).

В блок try, перед циклом для відліку часу, додаємо наступний код:

```
with open('coord.txt', 'w') as file:
    file.write(f'{gcj02_lng_lat[1]},{gcj02_lng_lat[0]}')
file.close()
```

Даний блок коду створює текстовий файл, в який записуються поточні координати платформи, необхідні для роботи майбутніх проектів на базі даної роботизованої платформи.

4.2.2 Налаштування передачі координат

Так само, як і у випадку з модулем GNSS, виробник модуля LoRa пропонує набір бібліотек та скрипт для взаємодії з модулем, що знову ж таки допоможе нам.

Вводимо в Command shell наступні команди та завантажуємо демо-версію коду для модуля LoRa від виробника:

```
cd Documents
wget
https://files.waveshare.com/upload/1/18/SX126X_LoRa_HAT_CODE.zip
unzip SX126X_LoRa_HAT_CODE.zip
```

Копіюємо з папки SX126X_LoRa_HAT_Code/raspberrypi/python/ усі файли, окрім main.py, до папки з роботою. В файлі sx126x.py шукаємо рядок зі змінною start_freq та задаємо її рівною 433 – частота в МГц, на якій працює модуль. Таким чином ми додали в папку роботи готові бібліотеки від виробника для роботи з інтерфейсом модуля LoRa. За замовчуванням, файли адаптовані для роботи з модулем LoRa через порт ttyS0. Так як модуль встановлений на роз'єми GPIO, то нічого змінювати не треба. В результаті, робоча тека має такий вигляд:

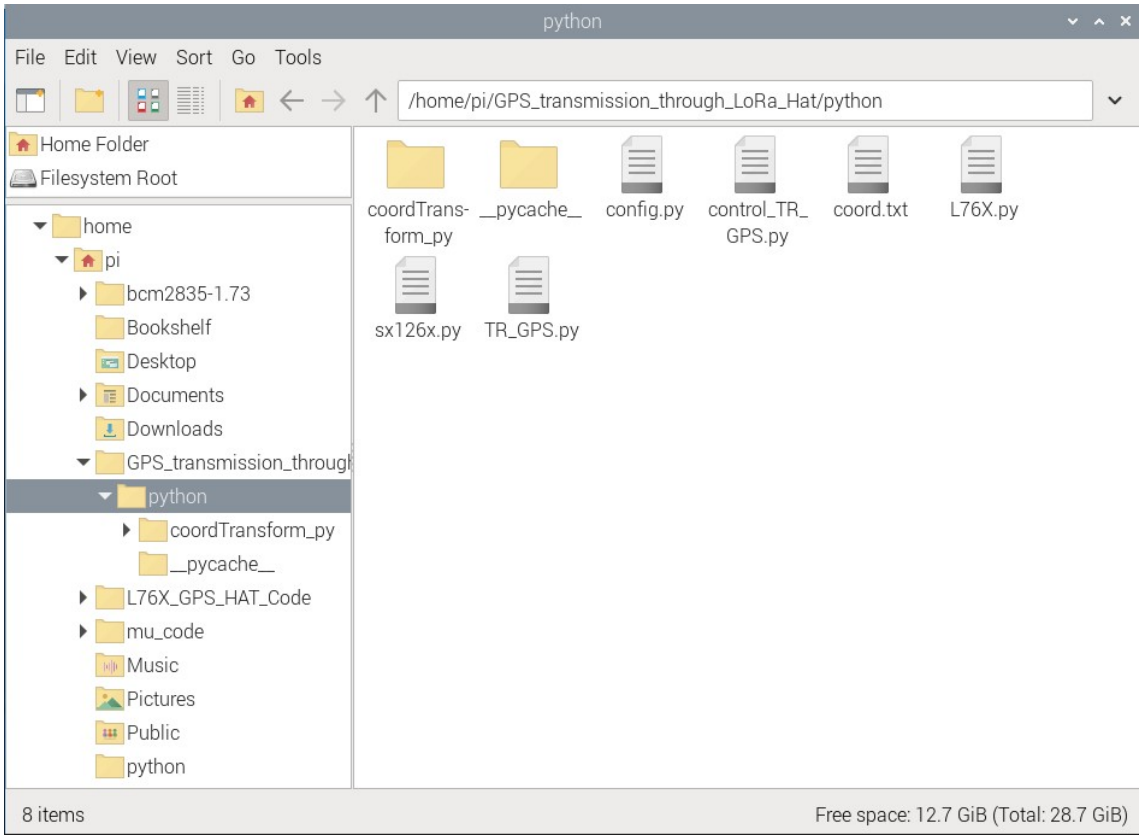

Рисунок 4.2.2.1 – Робоча тека ПЗ роботизованої платформи

Далі, відкриваємо в Thonny файл TR_GPS.py й додаємо до нього скрипт для передачі координат через радіоканал LoRa.

В список імпортованих бібліотек додамо наступні:

```
import sys
import sx126x
import threading
import select
import tty
```

Даний модуль LoRa працює за приватним протоколом передачі. Для налаштування каналу необхідно кожному модулю задати номер вузла (node). Згідно коментарів в sx126x.py, node може мати номер від 0 до 65535. Причому, node модуля приймача має бути більшим ніж передатчика.

Перед блоком try в коді додамо ще декілька рядків:

```
addrSERV = 65535
offset_frequence = int(433)-410
nodeL = sx126x.sx126x(serial_num =
"/dev/ttyS0",freq=433,addr=20,power=22,rssi=True,air_speed=2400,
relay=False
```

Змінна addrSERV – це адреса шлюзу LoRa встановленого на сервер, offset_frequence – параметр необхідний для формування повідомлення, МГц, nodeL – набір параметрів передатчика LoRa, freq – частота на який працює канал, МГц, addr – адреса модуля передавача LoRa, power – потужність випромінюваного сигналу, дБм, air_speed – швидкість передачі, Кб/с.

Тепер, в самому блоці try, одразу після блоку запису координат в текстовий файл, реалізуємо блок формування та надсилання повідомлення з координатами. Перш за все, за допомогою наступного рядка програма створює тимчасову змінну типу рядок, що містить в собі символьний запис з останніми отриманими координатами платформи:

```
get_t=f'{gcj02_lng_lat[1]},{gcj02_lng_lat[0]}'
```

Далі, формуємо саме повідомлення, яке буде надсилатися на шлюз LoRa. Його формат взятий з демо-версії коду виробника модуля. В ньому міститься інформація про адрес, куди буде відправлене повідомлення, частота каналу зв’язку, адреса відправника та самі координати:

```
data = bytes([int(addrSERV)>>8]) + bytes([int(addrSERV)&0xff]) +
bytes([offset_frequence]) + bytes([nodeL.addr>>8]) +
bytes([nodeL.addr&0xff]) + bytes([nodeL.offset_freq]) +
get_t.encode()
```

Даний короткий блок визначає та виводить в байтах інформацію про розмір повідомлення, необхідну для аналізу роботи системи:

```
dataSize=sys.getsizeof(data)
print(dataSize)
```

Нарешті, наступний рядок надсилає на модуль LoRa команду передати повідомлення з координатами на шлюз сервера:

```
nodeL.send(data)
```

Результат роботи створеної програми ми можемо перевірити запустивши в Command shell скрипт TR_GPS.py (рис. 4.2.2.2).

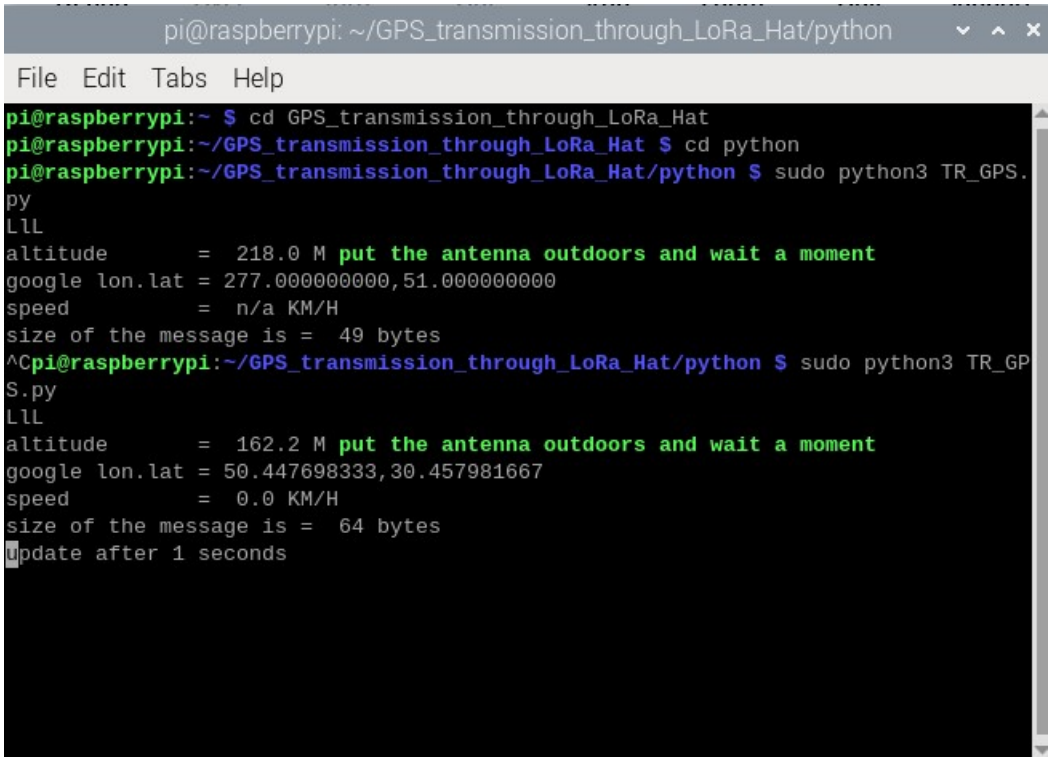

Рисунок 4.2.2.2 – Результат роботи програми TR_GPS.py

Як бачимо, дана програма отримує координати про місцеположення роботизованої платформи, виводить у вікно Command shell інформацію про висоту відносно рівня моря, координати, швидкість руху платформи та розмір надісланого на шлюз повідомлення.

На даному етапі, складова системи, відповідальна за визначення та надсилання координат роботизованої платформи повністю готова. Наступним кроком буде реалізація прийому та обробки координат.

4.3 Налаштування сервера

Для підключення та налаштування мікрокомп'ютера Raspberry Pi сервера, виконаємо ті ж самі дії як і в розділі 4.2.

4.3.1 Налаштування прийому сигналу LoRa

Встановлюємо на сервер те саме запропоноване виробником модуля LoRa ПЗ, що і на роботизованій платформі. Виконуємо в Command shell команди:

```
cd Documents
wget
https://files.waveshare.com/upload/1/18/SX126X_LoRa_HAT_CODE.zip
unzip SX126X_LoRa_HAT_CODE.zip
```

За адресою cd Documents/SX126X_LoRa_HAT_Code/raspberrypi/python знаходимо та відкриваємо в редакторі файл sx126x.py шукаємо рядок зі змінною start_freq та задаємо її рівною 433 – частота в МГц, на якій працює модуль.

На 57 рядку в файлі main.py знаходимо рядок `node = ...`, задаємо частоту freq = 433, адресу порту addr = 65535, все інше залишаємо як є.

Тепер запустимо програму для модуля LoRa main.py на сервері і перевіримо працездатність встановленого радіоканалу LoRa між передавачем роботизованої платформи та шлюзом сервера. Відкриємо Command shell і виконаємо команди:

```
cd ~/Documents/SX126X_LoRa_HAT_Code/raspberrypi/python/
sudo python3 main.py
```

Паралельно запускаємо на роботизованій платформі скрипт TR_GPS.py. Якщо все зроблено вірно, то у вікні command shell ми побачимо як відбувається прийом повідомлення від платформи (рис. 4.3.1.1).

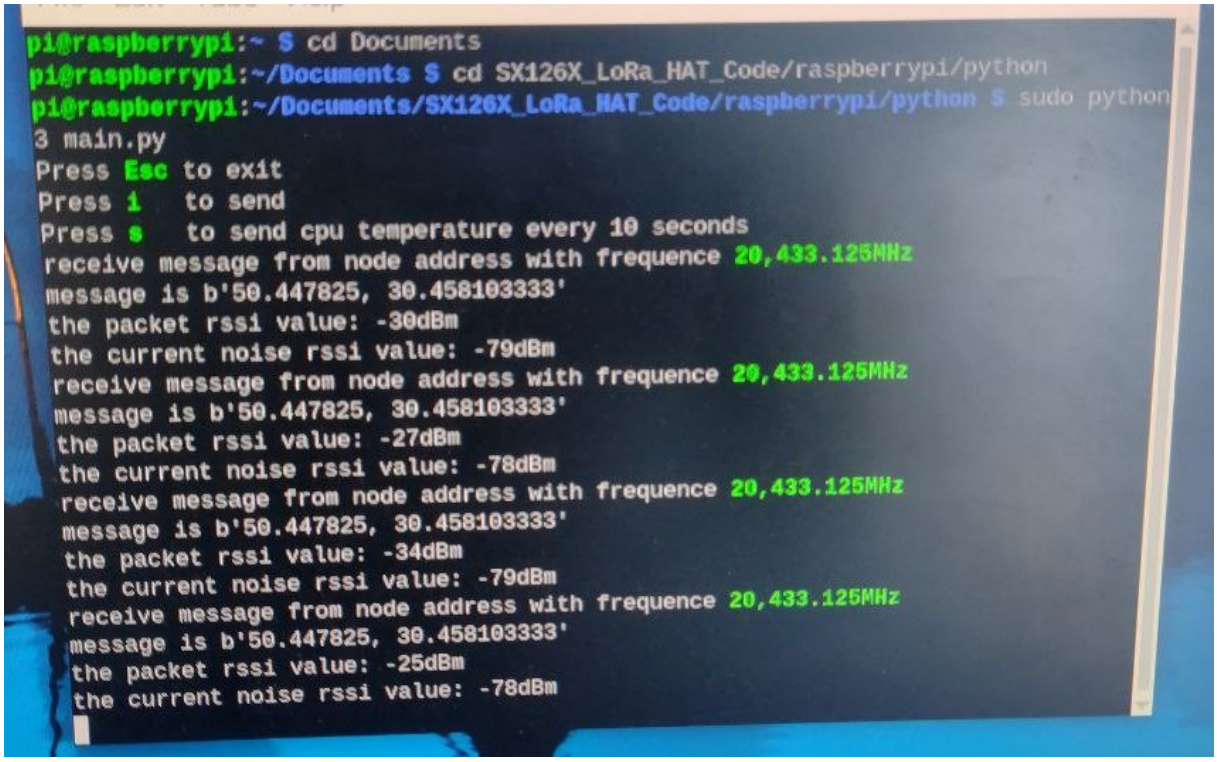

Рисунок 4.3.1.1 – Робота програми отримання повідомлень від передавача платформи на шлюзі сервера

Як бачимо на рис. 4.3.1.1, дана програма успішно отримала та прочитала повідомлення від платформи, вивела його на екран. Сам сигнал LoRa, що несе повідомлення з координатами платформи зображений на рисунку 4.3.1.2.

Рисунок 4.3.1.2 – Сигнал LoRa від передавача з повідомленням про координати платформи прийнятий на вході SDR приймача

4.3.2 Налаштування бази даних SQLite

Для роботи з базами даних встановимо SQLite на сервер на базі Raspberry Pi. В Command shell виконаємо наступну команду:

```
sudo apt install sqlite3
```

В теці Documents створюємо папку для файлів сервера. В цій папці створюємо базу даних locations та налаштовуємо її в оболонці SQLite. В Command shell водимо та виконуємо команди:

```
cd Documents
mkdir rpiWebServer
cd rpiWebServer
sqlite3 locations.db
```

Створимо таблицю, в якій будуть зберігатися отримані записи про координати платформи та потужності отриманого шлюзом сигналу. Таблиця містить в собі 4 колонки – автоматично виданий послідовний номер запису id, значення широти, довготи та потужності прийнятого сигналу. Виконаємо наступний блок команд в оболонці SQLite:

```
CREATE TABLE locations (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    latitude REAL NOT NULL,
    longitude REAL NOT NULL,
    rssi INTEGER NOT NULL
);
```

Після створення таблиці виходимо з оболонки SQLite:

```
.quit
```

На цьому, налаштування самої бази даних завершено. Тепер відкриваємо в редакторі Thonny файл main.py за адресою Documents/SX126X_LoRa_HAT_Code/raspberrypi/python і створимо в ньому блок додавання нових записів координат до бази даних. Шукаємо в коді оголошення функції `def receive(self)` і вставляємо в кінець функції блок коду, який буде додавати в базу даних нові записи про за допомогою інструменту conn:

```
with open('coord.txt', 'w') as file:
                dirtyWcoord=format(r_buff[3:-1])
                tempWcoord=dirtyWcoord.strip("b'")
                tempWrssi=format(256-r_buff[-1:][0])
                tempW=tempWcoord +', -'+ tempWrssi
                file.write(tempW)
            file.close()

            with
open('/home/pi/Documents/SX126X_LoRa_HAT_Code/raspberrypi/python
/coord.txt', 'r') as file:
                get_by_LoRa_data=file.read()
                [res_lat, res_lon,
res_rssi]=get_by_LoRa_data.split(',')
                db_path =
'/home/pi/Documents/rpiWebServer/locations.db'
                conn = sqlite3.connect(db_path)
                cursor = conn.cursor()
                cursor.execute('''
                    INSERT INTO locations (latitude, longitude,
rssi)
                    VALUES (?, ?, ?)
                ''', (res_lat, res_lon, res_rssi))
                conn.commit()
                conn.close()
          file.close()
```

Даний код працює за наступним алгоритмом: створюється та відкривається файл .txt, в якому будуть зберігатися дані про останні отримані координати платформи, з отриманого повідомлення видобуваються координати та rssi сигналу, ці дані форматуються та записуються у текстовий файл. Далі, ці дані зчитуються, налаштовується підключення до бази даних і за допомогою conn.cursor в базу даних вноситься новий запис. Після цього, файл і база даних закриваються і робота даного блоку коду завершується.

4.3.3 Налаштування веб-сторінки на Flask

Перш за все, треба встановити Flask на Raspberry Pi. Відкриємо Command shell та виконаємо:

```
sudo apt install python3-flask
```

Перейдемо в теку rpiWebServer і створимо в ній теки static і templates:

```
cd Documents
cd rpiWebServer
mkdir static
mkdir templates
```

В теці rpiWebServer створимо файл app.py, який буде запускати нашу веб-сторінку. В теці templates створимо файл index.html, в якому буде налаштована структура веб-сторінки. В теці static створимо файл style.css з налаштуваннями кольору елементів веб-сторінки. Фінальний вигляд теки:

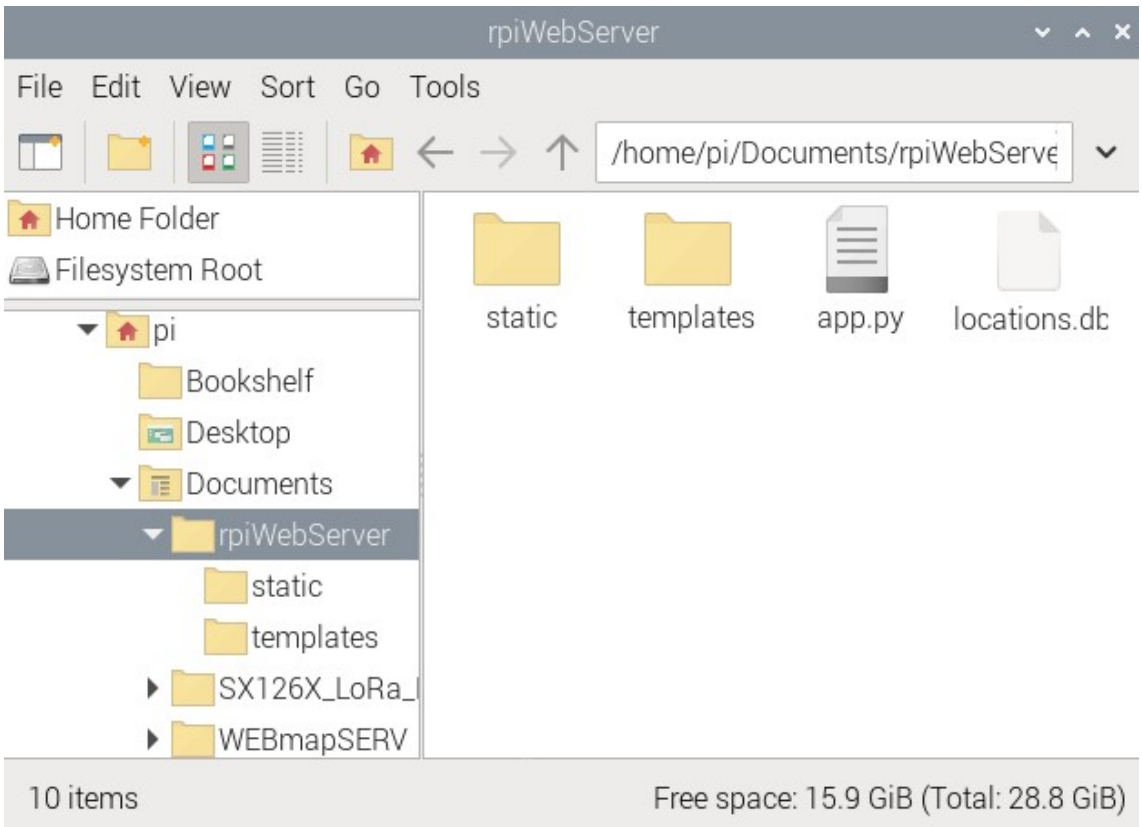

Рисунок 4.3.3.1 – Робоча тека ПЗ веб-сторінки сервера

Відкриваємо в редакторі Thonny файл app.py і приступаємо до створення програми веб-сторінки.

Включаємо в проект необхідні бібліотеки та модуль Flask:

```
from flask import Flask, jsonify, render_template
import datetime
import sqlite3
```

Створюємо об’єкт Flask під назвою app:

```
app = Flask(__name__)
```

Оголошуємо функцію для підключення до бази даних location.db:

```
def get_db_connection():
    conn = sqlite3.connect('locations.db')
    conn.row_factory = sqlite3.Row
    return conn
```

Оголошуємо функцію для видобування записів про координати та rssi з бази даних та передачі для відображення на мапі:

```
@app.route('/api/locations')
def get_locations():
    conn = get_db_connection()
locations = conn.execute('SELECT * FROM locations').fetchall()
    conn.close()

    return jsonify([dict(ix) for ix in locations])
@app.route('/')
```

Оголошуємо функцію index. В неї входить отримання поточних дати і часу на сервері, видобування останніх отриманих координат роботизованої платформи з текстового файлу в теці програми шлюзу LoRa та відправка отриманих даних в index.html для відображення на веб-сторінці:

```
def index():
    now = datetime.datetime.now()
    timeString = now.strftime("%Y-%m-%d %H:%M")
    with
open('/home/pi/Documents/SX126X_LoRa_HAT_Code/raspberrypi/python
/coord.txt', 'r') as file:
        get_by_LoRa_data=file.read()
        [res_lat, res_lon, res_rssi]=get_by_LoRa_data.split(',')
    file.close()

    templateData = {
        'title' : 'Coordinates of robotic platform',
```

```
        'time': timeString,
        'coord': (res_lat +', ' + res_lon)
    }

    return render_template('index.html', **templateData)
```

Завершуємо код програми визначення адресу та порту веб-сторінки а активацією контролю виникнення помилок на сервері:

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80, debug=True)
```

Зберігаємо та закриваємо app.py. Заходимо в папку templates і відкриваємо в редакторі Thonny файл index.html. Оголошуємо тип файлу, заголовок та підключаємо зовнішні стилі:

```
<!DOCTYPE html>
</html>
    <head>
        <title>{{ title }}</title>
        <link rel="stylesheet" href="../static/style.css/">
        <link rel="stylesheet"
href="https://unpkg.com/leaflet/dist/leaflet.css" />
        <style>
            #map {
                height: 700px;
                width: 100%;
            }
        </style>
    </head>
```

Оголошуємо тіло сторінки, вивід дати і часу на сервері, відображення останніх отриманих координат платформи та легенди мапи:

```
<body>
        <h1>Hello, World!</h1>
        <h2>The date and time on the server is: {{ time }}</h2>
        <h3>The latest coordinates of CAR is: {{ coord }}</h3>
        <h4>Blue dot - location of LoRa reciver, Green dot -
RSSI > -55 dBm, Yellow dot - RSSI > -70 dBm, Orange dot - RSSI >
-85 dBm, Red dot - RSSI < -85 dBm, under noises</h4>
        <h5>Cyan circule - 5 km radius, Red circule - 0.9 km
radius,</h5>
```

Додаємо на веб-сторінку мапу з ресурсу Open Street Map:

```
<div id="map"></div>
<script
src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
<script>
     var map = L.map('map').setView([50.4,30.6], 16);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png'
, {maxZoom: 19,}).addTo(map);
```

Додаємо на мапу коло, що відображає потенційну дальність зв’язку між двома LoRa:

```
L.circle([50.4,30.6], {
     color: 'cyan',
     fillColor: 'cyan',
     fillOpacity: 0.1,
     radius: 5000
}).addTo(map);
```

Додаємо алгоритм, що визначає колір розміщеної за координатами з бази даних точки в залежності від потужності rssi запису координат:

```
fetch('/api/locations')
                .then(response => response.json())
                .then(data => {
                    data.forEach((location, index) => {
                        let color;
                        if (location.rssi === 0) {
                            color = 'blue';
                        }
                        else if (index === data.length - 1) {
                        color = 'white';   // Last point
                        } else {
                            if (location.rssi >= -55) {
                                color = 'green';
                            } else if (location.rssi >= -70) {
                                color = 'yellow';
                            } else if (location.rssi >= -85) {
                                color = 'orange';
                            } else {
                                color = 'red';
                            }
                        }
```

Додаємо алгоритм створення точок на мапі та відображення їх rssi та закриваємо скрипт, тіло документа та сам блок html:

```
   let marker = L.circleMarker([location.latitude,
location.longitude], {
                                    color: color,
                                    radius: 4,
                                    fillOpacity: 0.8
                                }).addTo(map);
                                marker.bindPopup(`RSSI:
${location.rssi}`);
                            });
                        });
            </script>
        </body>
</html>
```

Зберігаємо та закриваємо файл.

На виході ми маємо веб-сторінку з повністю розробленим бек-енд та фронт-енд складовими. В Command shell переходимо в директорію веб-сторінки та виконуємо скрипт app.py. При введені в браузері ip-адреси сервера Raspberry Pi з вказівкою на 80 порт, відкриваємо веб-сторінку сервера (рис. 4.3.3.2).

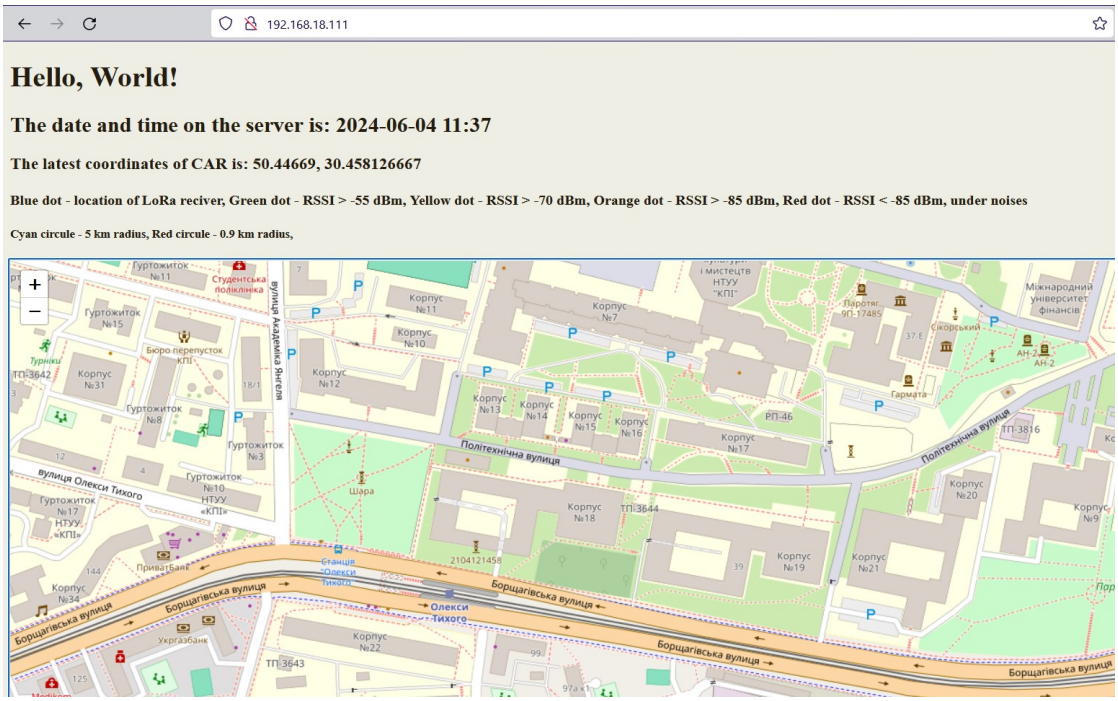

Рисунок 4.3.3.2 – Веб-сторінка сервера

5 ОПИС ТА РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ

5.1 Хід експерименту

Експериментальне дослідження проводилося два дні – 29 та 30 травня 2024 року. Завданням експерименту було дослідити реальну дальність зв’язку за допомогою модулів LoRa між роботизованою платформою та сервером. Сервер знаходився на висоті 25 метрів від рівня, на якому проходив маршрут переміщення платформи, біля вікна всередині приміщення. Експеримент проводився за сонячної погоди в одному зі спальних районів міста Київ, де були присутні як і великі відкриті території, так і щільна 9-, 16- та 24-поверхова забудова. Параметри сигналу LoRa, яким передавалися координати роботизованої платформи вказані в таблиці 5.1.1.

Таблиця 5.1.1 – Параметри сигналу LoRa

Несуча частота сигналу LoRa	433,125 МГц
Ширина сигналу	0,5 МГц
Довжина повідомлення	64 Байт
Швидкість передачі	2400 Біт/с
Потужність випромінюваного сигналу	22 дБм
Періодичність надсилання сигналу	10 с

Під час проведення експериментів, на роботизованій платформі була запущена програма TR_GPS.py, що передавала координати на сервер. На сервері працювали програма шлюзу LoRa для прийому повідомлень main.py та app.py для роботи веб-сторінки.

В ході експерименту, роботизовану платформу переміщували такими маршрутами (рис. 5.1.1), щоб охопити максимальну кількість типів середовищ поширення сигналу – відкрита місцевість, низька забудова,

щільна висотна забудова. Сигнал дійшов навіть з ліфтової шахти будівлі, в якій розміщувався сервер. Максимальна відстань, на яку платформа була віддалена від сервера – 920 метрів.

Рисунок 5.1.1 – Маршрути переміщення роботизованої платформи: синім – 29 травня, червоним – 30 травня

В результаті, за два дні дослідження, вдалося зібрати понад 600 записів координат платформи та rssi сигналу. Таким чином, маємо достатньо матеріалу для аналізу дальності зв’язку розробленої системи.

Уривок з історії отриманих шлюзом повідомлень під час проведення експерименту 30 травня 2024 року наведено у додатку Б.

5.2 Обробка отриманих результатів

Як вже зазначалося, в ході експерименту вдалося зробити близько 600 записів. Проте, веб-сторінка має обмеження в 256 об’єктів, які вона може відобразити на мапі. Тому потрібно провести попередню обробку.

Провівши аналіз історії отриманих повідомлень (додаток Б) та переглянувши на веб-сторінці мапу з попередніми результатами дослідження бачимо, що велику частину отриманих координат складають хибні точки (рис. 5.2.1), які не несуть жодної користі у репрезентації дальності зв’язку розробленої системи передачі координат і тому їх можна прибрати.

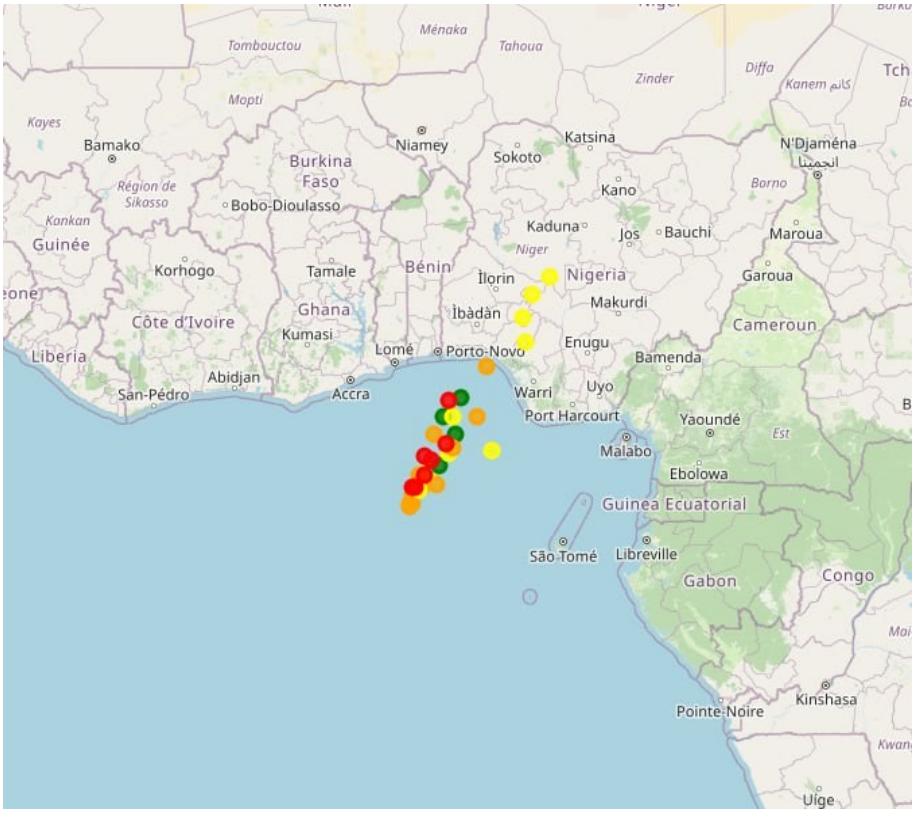

Рисунок 5.2.1 – Хибні координати

Частину хибних координат, що за значеннями перевалили за 9-тисячну координату, складають записи, отримані під час знаходження платформи всередині будівлі, де важко зловити достатню кількість сигналів від супутників та отримати точні координати. Іншу частина хибних координат викликана помилками в роботі модуля GNSS, і вказують що платформа знаходиться в районі нульових координат.

В усіх цих похибок є одна спільна риса. Так як експеримент проводився в Києві, то відомо, що приблизні координати роботизованої платформи обов’язково будуть 50 градусів північної широти та 30 градусів східної довготи. Тому, можна провести видалення хибних координат за цією ознакою.

На сервері відкриваємо Command shell та відкриваємо базу даних з результатами експериментів в оболонці SQLite:

```
cd Documents
cd rpiWebServer
sqlite3 locations.db
```

Скористаємося фільтрацією записів за значенням широти в 50 градусів. Використаємо для цього команду sqlite DELATE:

```
DELETE FROM locations WHERE latitude NOT LIKE '50%';
```

Так як переміщення платформи часто зупинялося, на сервер могли надходити дані з однаковими координатами. Їх також можна прибрати за допомогою сортування та впорядкування бази даних. Для цього можна створити тимчасову таблицю всередині нашої бази даних і додати в неї відсортовані записи. Виконаємо наступні команди в оболонці SQLite:

```
CREATE TABLE unique_locations (
     id INTEGER PRIMARY KEY AUTOINCREMENT,
     latitude REAL NOT NULL,
     longitude REAL NOT NULL,
     rssi INTEGER NOT NULL
);
INSERT INTO unique_locations (latitude, longitude, rssi)
SELECT latitude, longitude, rssi
FROM locations
GROUP BY latitude, longitude;
```

Далі скидаємо стару таблицю, перейменовуємо нову таблицю та перевіряємо кількість записів:

```
DROP TABLE locations;
ALTER TABLE unique_locations RENAME TO locations;
SELECT * FROM locations;
.quit
```

Тепер, після усіх пройдених етапів фільтрації даних, з понад 600 записів лишилось тільки 260 унікальних записів, яких цілком вистачить для аналізу дальності роботи системи. Переглянути перелік записів обробленої бази даних експерименту можна в додатку В.

5.3 Аналіз отриманих результатів

В результаті проведеного експериментального дослідження, вдалося отримати та відобразити на мапі веб-сторінки сервера дані про залежність потужності сигналу від відстані між двома модулями LoRa (рис. 5.3.1).

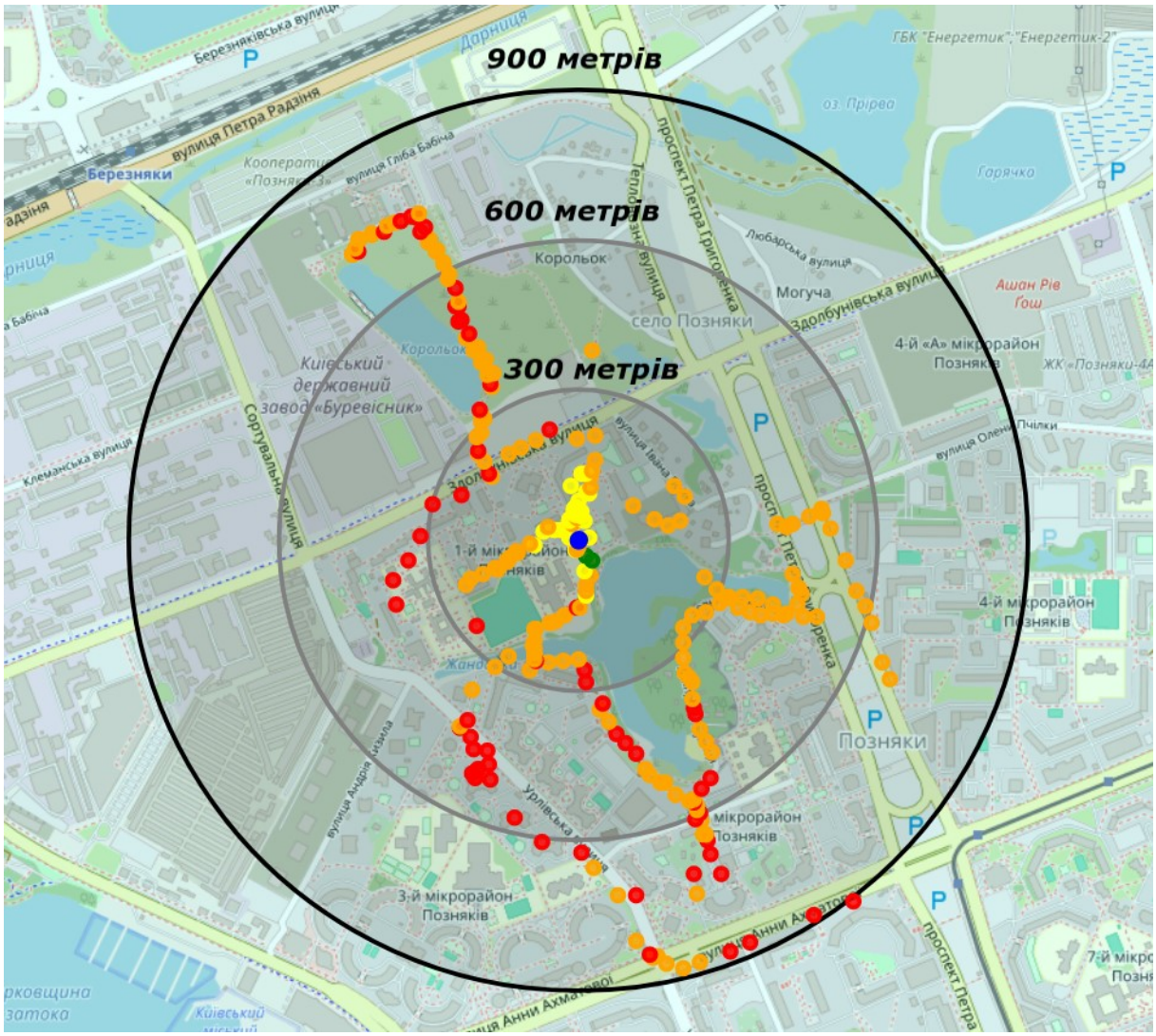

Рисунок 5.3.1 – Графік потужності сигналу роботизованої платформи на шлюзі в залежності від відстані

Легенда до мапи на рис. 5.3.1: залежність кольору точки від потужності надісланого з даної точки сигналу:

- – розташування шлюзу LoRa
- – потужність > -55 дБм
- – потужність > -70 дБм
- – потужність > -85 дБм
- – потужність < -85 дБм, сигнал схований в шумах

Чорні та сірі концентричні відносно розташування сервера кола на мапі необхідні для зручного орієнтування у відстанях до сервера.

Згідно мапи бачимо, що відстань від сервера до найвіддаленішої точки, з якої сигнал від роботизованої платформи надійшов на шлюз, трохи більше 900 метрів, а rssi сигналу – -87 дБм (рис. 5.3.2).

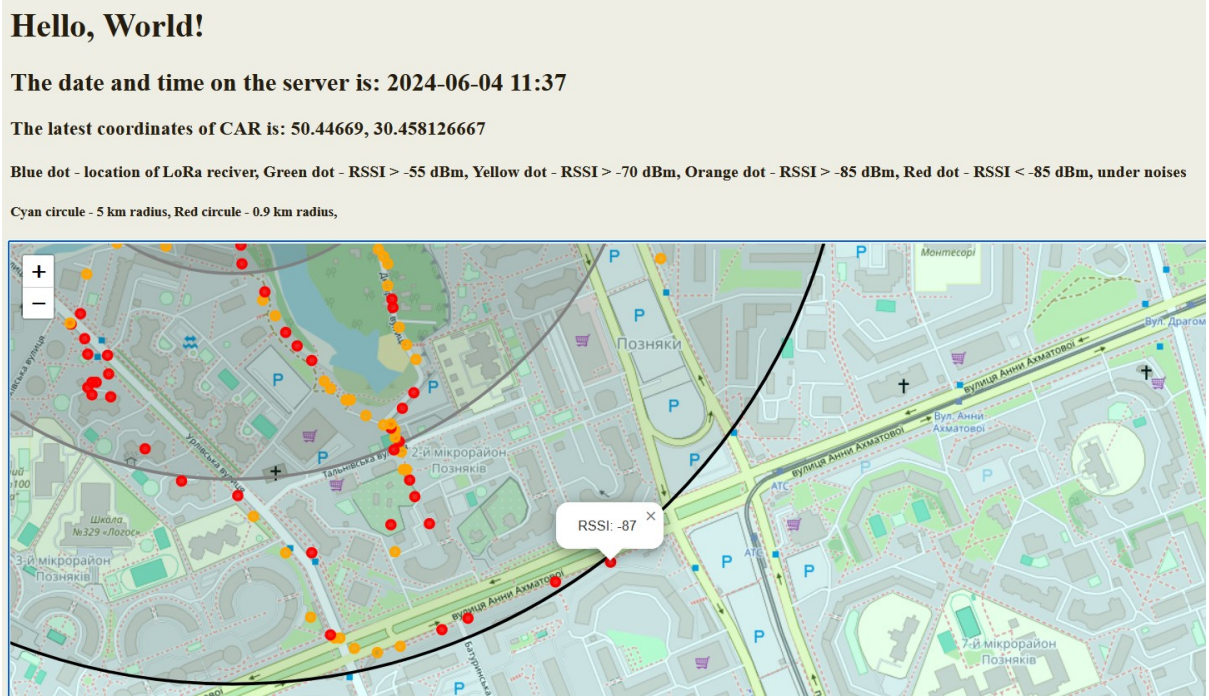

Рисунок 5.3.2 – RSSI найвіддаленішої від сервера точки

Проаналізувавши історію отриманих шлюзом повідомлень, можна сказати, що, в середньому, рівень завад тримається на позначці -85 дБм. Найслабкіший отриманий сигнал мав rssi -89 дБм. Отже, в ході експерименту

встановлено, що дана система на базі модулів LoRa здатна відрізняти сигнали, що на 4 дБ нижче рівня шумів.

Якщо порівняти між собою рисунки 5.1.1 та 5.3.1, то можна помітити, що з деяких ділянок маршруту сигнал від платформи не дійшов до сервера. Відобразимо ці сліпі зони на рис. 5.3.3 та проаналізуємо причини їх появи.

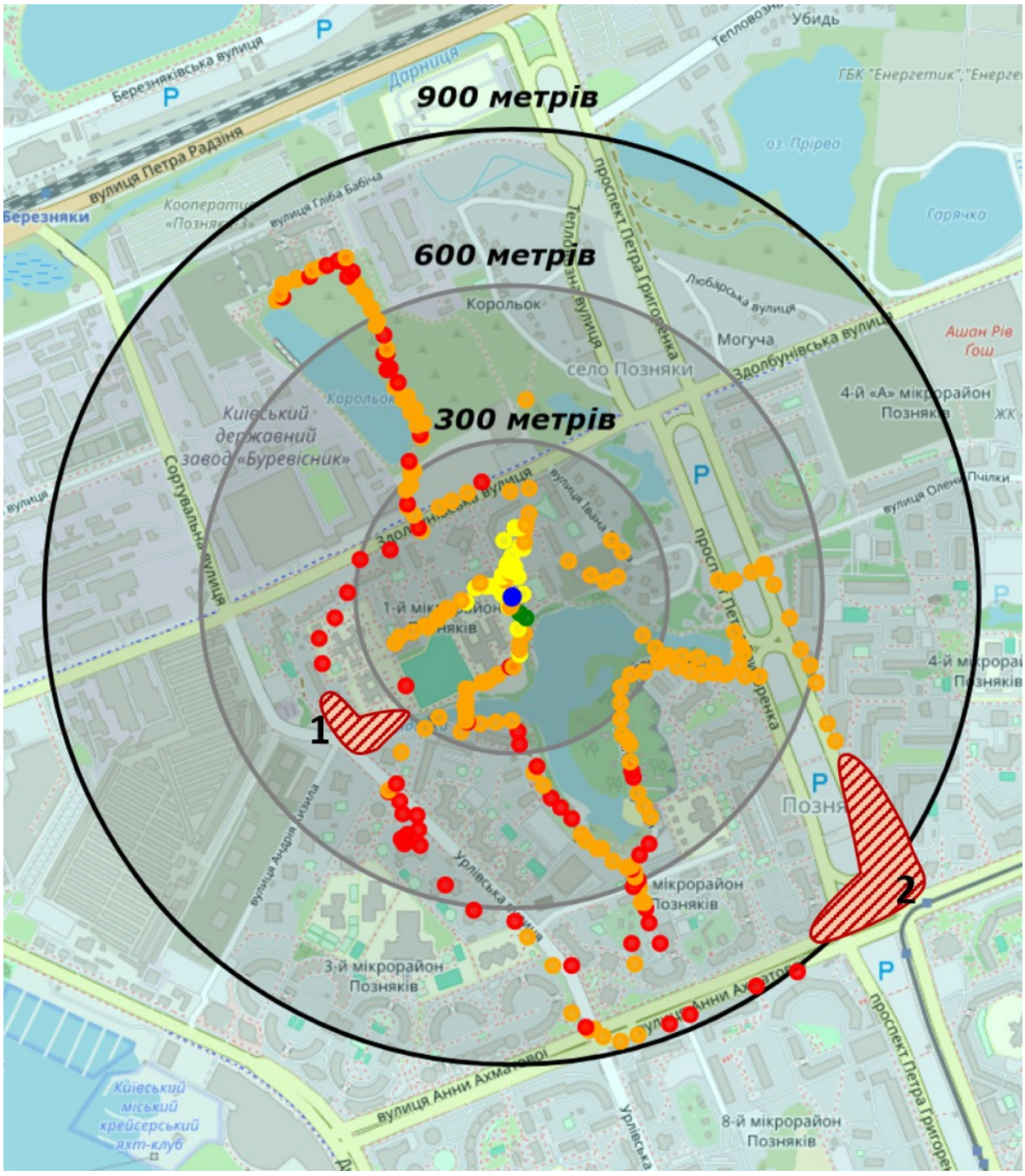

Рисунок 5.3.3 – Сліпі зони (заштриховані червоним полігони) маршруту, з яких сигнал не дійшов до шлюзу

Можна припустити, що причиною наявності сліпої зони 1 є те, що вона розташована паралельно площині вікна, за яким стоїть шлюз сервера. І враховуючи відстань між сервером та даною сліпою зоною, цілком імовірно що сигналу не вистачило потужності оминути торець будинку і потрапити на шлюз LoRa.

Між сліпою зоною номер 2 та шлюзом знаходиться ряд щільно розміщених багатоповерхівок. Враховуючи, що відстань між даною зоною та сервером майже в районі граничної дальності роботи системи, роботизована платформа знаходиться на рівні землі, шлюз на висоті 25 метрів, а висота багатоповерхівок понад 50 м. Сигнал просто не зміг обігнути перешкоду.

З цього можна зробити висновок, що для зменшення кількості сліпих зон, шлюз LoRa треба розташовувати якомога вище над потенційними бар’єрами і так, щоб в усі сторони можна було встановити зв'язок по прямій видимості.

Як бачимо на рис. 5.3.3 та 5.3.1, перші 100-200 метрів потужність сигналу різко спадає, через що на мапі вже починаючи з 250 метрів більше всього точок в зоні біля шумів, що в свою чергу важливо розуміти, на яких відстанях сигнал почне ховатися в шумах.

ВИСНОВКИ

В даній дипломній роботі було розроблено систему передачі координат роботизованої платформи з використанням технології LoRa, зібрано експериментальний макет системи, що включає в себе макет роботизованої платформи та сервера зі шлюзом LoRa, налаштовано взаємодію складових системи між собою, встановлено радіоканал зв’язку між ними, розроблено програмне забезпечення для функціонування системи, створено веб-сайт для відображення результатів дослідження, проведено дослідження дальності зв’язку системи. Були розглянуті принципи роботи технології LoRa, системи GNSS, проаналізовано ринок мікрокомп’ютерів та модулів GNSS і LoRa для них.

В ході експерименту було встановлено, що дана система здатна підтримувати зв’язок для передачі координат між роботизованою платформою та сервером на відстанях до 900 метрів в умовах щільної висотної міської забудови за рівня шумів до -79 дБм в радіоканалі. Шлюз зміг розрізнити сигнал з потужністю до 4 дБ нижче рівня шумів, проти очікуваних 20 дБ.

Можливим рішенням для поліпшення дальності зв’язку системи є зменшення швидкості передачі, що звужує сигнал та підвищує чутливість приймача. Але навіть характеристик системи на даному етапі достатньо, для реалізації локальних IoT проектів (наприклад на території кампусу КПІ) з участю роботизованих платформ з системою зв’язку через LoRa.

За результатами роботи була зроблена доповідь на науковій конференції, тези якої опубліковані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Набір з можливостями машинного навчання PiRacer AI Kit. [Електронний ресурс]. Доступно за посиланням: https://www.waveshare.com/wiki/PiRacer_AI_Kit. Останній вхід 13.06.2024.

2. Електроживлення мікрокомп’ютерів Raspberry Pi (EN). [Електронний ресурс]. Доступно за посиланням: https://github.com/raspberrypi/documentation/blob/develop/documentation/asciidoc/computers/raspberry-pi/power-supplies.adoc. Останній вхід 13.06.2024.

3. Плата розширення L76X GPS. [Електронний ресурс]. Доступно за посиланням: https://www.waveshare.com/wiki/L76X_GPS_HAT. Останній вхід 13.06.2024.

4. Плата розширення SX1268 433M LoRa. [Електронний ресурс]. Доступно за посиланням: https://www.waveshare.com/wiki/SX1268_433M_LoRa_HAT. Останній вхід 13.06.2024.

ДОДАТКИ

Додаток А

Лістинг програм

Програми сервера:

app.py

```
from flask import Flask, jsonify, render_template
import datetime
import sqlite3

app = Flask(__name__)

def get_db_connection():
    conn = sqlite3.connect('locations.db')
    conn.row_factory = sqlite3.Row
    return conn

@app.route('/api/locations')
def get_locations():
    conn = get_db_connection()
    locations = conn.execute('SELECT * FROM
locations').fetchall()
    conn.close()

    return jsonify([dict(ix) for ix in locations])

@app.route('/')
def index():
    now = datetime.datetime.now()
    timeString = now.strftime("%Y-%m-%d %H:%M")
    with
open('/home/pi/Documents/SX126X_LoRa_HAT_Code/raspberrypi/python
/coord.txt', 'r') as file:
        get_by_LoRa_data=file.read()
        [res_lat, res_lon, res_rssi]=get_by_LoRa_data.split(',')
    file.close()

    templateData = {
        'title' : 'Coordinates of robotic platform',
        'time': timeString,
        'coord': (res_lat +', ' + res_lon)
    }
    return render_template('index.html', **templateData)
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80, debug=True)
```

/templates/index.html

```
<!DOCTYPE html>
</html>
    <head>
        <title>{{ title }}</title>
        <link rel="stylesheet" href="../static/style.css/">
        <link rel="stylesheet"
href="https://unpkg.com/leaflet/dist/leaflet.css" />
        <style>
            #map {
                height: 700px;
                width: 100%;
            }
        </style>
    </head>

    <body>
        <h1>Hello, World!</h1>
        <h2>The date and time on the server is: {{ time }}</h2>
        <h3>The latest coordinates of CAR is: {{ coord }}</h3>
        <h4>Blue dot - location of LoRa reciver, Green dot -
RSSI > -55 dBm, Yellow dot - RSSI > -70 dBm, Orange dot - RSSI >
-85 dBm, Red dot - RSSI < -85 dBm, under noises</h4>
        <h5>Cyan circule - 5 km radius, Red circule - 0.9 km
radius,</h5>
        <div id="map"></div>
        <script
src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
        <script>
            var map = L.map('map').setView([50.41438,30.61855],
16);


L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png'
, {
                maxZoom: 19,
            }).addTo(map);

            L.circle([50.41438,30.61855], {
                color: 'cyan',
                fillColor: 'cyan',
```

```
            fillOpacity: 0.1,
            radius: 5000
        }).addTo(map);

        L.circle([50.41438,30.61855], {
            color: 'red',
            fillColor: '#f03',
            fillOpacity: 0.1,
            radius: 900
        }).addTo(map);


        fetch('/api/locations')
            .then(response => response.json())
            .then(data => {
                data.forEach((location, index) => {
                    let color;
                    if (location.rssi === 0) {
                        color = 'blue';
                    }
                    else if (index === data.length - 1) {
                    color = 'white';   // Last point
                    } else {
                        if (location.rssi >= -55) {
                            color = 'green';
                        } else if (location.rssi >= -70) {
                            color = 'yellow';
                        } else if (location.rssi >= -85) {
                            color = 'orange';
                        } else {
                            color = 'red';
                        }
                    }

                    let marker =
L.circleMarker([location.latitude, location.longitude], {
                        color: color,
                        radius: 4,
                        fillOpacity: 0.8
                    }).addTo(map);
                    marker.bindPopup(`RSSI:
${location.rssi}`);

                });
            });
```

```
        </script>
    </body>
</html>
```

/home/pi/Documents/SX126X_LoRa_HAT_Code/raspberrypi/python/main.py

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

#
#    this is an UART-LoRa device and thers is an firmware on
Module
#    users can transfer or receive the data directly by UART and
dont
#    need to set parameters like coderate,spread factor,etc.
#    |========================================== |
#    |   It does not suport LoRaWAN protocol !!!   |
#    | ==========================================|
#
#    This script is mainly for Raspberry Pi 3B+, 4B, and Zero
series
#    Since PC/Laptop does not have GPIO to control HAT, it
should be configured by
#    GUI and while setting the jumpers,
#    Please refer to another script pc_main.py
#

import sys
import sx126x
import threading
import time
import select
import termios
import tty
from threading import Timer

old_settings = termios.tcgetattr(sys.stdin)
tty.setcbreak(sys.stdin.fileno())


#
#    Need to disable the serial login shell and have to enable
serial interface
#    command `sudo raspi-config`
```

```
#    More details: see
https://github.com/MithunHub/LoRa/blob/main/Basic%20Instruction.
md
#
#    When the LoRaHAT is attached to RPi, the M0 and M1 jumpers
of HAT should be removed.
#


#    The following is to obtain the temprature of the RPi CPU
def get_cpu_temp():
    tempFile = open( "/sys/class/thermal/thermal_zone0/temp" )
    cpu_temp = tempFile.read()
    tempFile.close()
    return float(cpu_temp)/1000

#   serial_num
#       PiZero, Pi3B+, and Pi4B use "/dev/ttyS0"
#
#    Frequency is [850 to 930], or [410 to 493] MHz
#
#    address is 0 to 65535
#        under the same frequence,if set 65535,the node can
receive
#        messages from another node of address is 0 to 65534 and
similarly,
#        the address 0 to 65534 of node can receive messages
while
#        the another note of address is 65535 sends.
#        otherwise two node must be same the address and
frequence
#
#    The tramsmit power is {10, 13, 17, and 22} dBm
#
#    RSSI (receive signal strength indicator) is {True or False}
#        It will print the RSSI value when it receives each
message
#

# node = sx126x.sx126x(serial_num =
"/dev/ttyS0",freq=433,addr=0,power=22,rssi=False,air_speed=2400,
relay=False)
node = sx126x.sx126x(serial_num =
"/dev/ttyS0",freq=433,addr=65535,power=22,rssi=True,air_speed=24
00,relay=False)
```

```
def send_deal():
    get_rec = ""
    print("")
    print("input a string such as \033[1;32m0,868,Hello
World\033[0m,it will send `Hello World` to lora node device of
address 0 with 868M ")
    print("please input and press Enter key:",end='',flush=True)

    while True:
        rec = sys.stdin.read(1)
        if rec != None:
            if rec == '\x0a': break
            get_rec += rec
            sys.stdout.write(rec)
            sys.stdout.flush()

    get_t = get_rec.split(",")

    offset_frequence = int(get_t[1])-(850 if int(get_t[1])>850
else 410)
    #
    # the sending message format
    #
    #         receiving node              receiving node
receiving node           own high 8bit           own low 8bit
own
    #         high 8bit address           low 8bit address
frequency                address                 address
frequency             message payload
    data = bytes([int(get_t[0])>>8]) +
bytes([int(get_t[0])&0xff]) + bytes([offset_frequence]) +
bytes([node.addr>>8]) + bytes([node.addr&0xff]) +
bytes([node.offset_freq]) + get_t[2].encode()

    node.send(data)
    print('\x1b[2A',end='\r')
    print(" "*200)
    print(" "*200)
    print(" "*200)
    print('\x1b[3A',end='\r')

def send_cpu_continue(continue_or_not = True):
    if continue_or_not:
        global timer_task
```

```
        global seconds
        #
        # boarcast the cpu temperature at 868.125MHz
        #
        data = bytes([255]) + bytes([255]) + bytes([18]) +
bytes([255]) + bytes([255]) + bytes([12]) + "CPU
Temperature:".encode()+str(get_cpu_temp()).encode()+"
C".encode()
        node.send(data)
        time.sleep(0.2)
        timer_task = Timer(seconds,send_cpu_continue)
        timer_task.start()
    else:
        data = bytes([255]) + bytes([255]) + bytes([18]) +
bytes([255]) + bytes([255]) + bytes([12]) + "CPU
Temperature:".encode()+str(get_cpu_temp()).encode()+"
C".encode()
        node.send(data)
        time.sleep(0.2)
        timer_task.cancel()
        pass

try:
    time.sleep(1)
    print("Press \033[1;32mEsc\033[0m to exit")
    print("Press \033[1;32mi\033[0m   to send")
    print("Press \033[1;32ms\033[0m   to send cpu temperature
every 10 seconds")

    # it will send rpi cpu temperature every 10 seconds
    seconds = 10

    while True:

        if select.select([sys.stdin], [], [], 0) ==
([sys.stdin], [], []):
            c = sys.stdin.read(1)

            # dectect key Esc
            if c == '\x1b': break
            # dectect key i
            if c == '\x69':
                send_deal()
            # dectect key s
            if c == '\x73':
```

```
                print("Press \033[1;32mc\033[0m   to exit the
send task")
                timer_task = Timer(seconds,send_cpu_continue)
                timer_task.start()

                while True:
                    if sys.stdin.read(1) == '\x63':
                        timer_task.cancel()
                        print('\x1b[1A',end='\r')
                        print(" "*100)
                        print('\x1b[1A',end='\r')
                        break

            sys.stdout.flush()

        node.receive()

        # timer,send messages automatically

except:
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN,
old_settings)
    # print('\x1b[2A',end='\r')
    # print(" "*100)
    # print(" "*100)
    # print('\x1b[2A',end='\r')

termios.tcsetattr(sys.stdin, termios.TCSADRAIN, old_settings)
# print('\x1b[2A',end='\r')
# print(" "*100)
# print(" "*100)
# print('\x1b[2A',end='\r')
```

Роботизована платформа:

/home/pi/GPS_transmission_through_LoRa_Hat/python/TR_GPS.py

```
# -*- coding:utf-8 -*-
import L76X
import time
import math
from gps3 import agps3
import coordTransform_py.coordTransform_utils as transform
import sys
import sx126x
import threading
import select
```

```
  import tty
  from threading import Timer

  x=L76X.L76X()
  x.L76X_Send_Command(x.SET_POS_FIX_400MS);

  #Set output message
  x.L76X_Send_Command(x.SET_NMEA_OUTPUT);

  x.L76X_Exit_BackupMode();

  #GPSDSocket creates a GPSD socket connection & request/retrieve GPSD output.
  gps_socket = agps3.GPSDSocket()
  #DataStream unpacks the streamed gpsd data into python dictionaries.
  data_stream = agps3.DataStream()
  gps_socket.connect()
  gps_socket.watch()

  gcj02_lng_lat = [0.0,0.0]            #buffer to save gcj02 coordinate

  addrSERV=65535
  offset_frequence = int(433)-410
  nodeL          =          sx126x.sx126x(serial_num          = "/dev/ttyS0",freq=433,addr=20,power=22,rssi=True,air_speed=2400, relay=False)

  try:
      print('LlL')
      for new_data in gps_socket:
          if new_data:
              data_stream.unpack(new_data)    #uppack the nmea
              if data_stream.lat != 'n/a' and data_stream.lon != 'n/a':                                        # if the data ready or not
                  gcj02_lng_lat                                    = transform.wgs84_to_gcj02(float(data_stream.lon),float(data_strea m.lat))                 # transform coordinate,wgs84 to gcj02
                  print('altitude                              =    ', data_stream.alt,'M',end='\r\n',flush=True)
# show the altitude
```

```
                print('google     lon.lat     =     %.9f,%.9f'
%(gcj02_lng_lat[1],gcj02_lng_lat[0]),end='\r\n',flush=True)
# show gcj02 coordinate,google map use wgs84 coordinate
                print('speed                                =   ',
data_stream.speed,'KM/H',end='\r\n',flush=True)
# show speed

                with open('coord.txt', 'w') as file:

file.write(f'{gcj02_lng_lat[1]},{gcj02_lng_lat[0]}')
                file.close()
                get_t=f'{gcj02_lng_lat[1]},{gcj02_lng_lat[0]}'
                data     =     bytes([int(addrSERV)>>8])     +
bytes([int(addrSERV)&0xff])    +    bytes([offset_frequence])    +
bytes([nodeL.addr>>8])      +      bytes([nodeL.addr&0xff])      +
bytes([nodeL.offset_freq]) + get_t.encode()
                nodeL.send(data)

                for          i          in          range(0,10):
# delay 10 seconds
                    print('update           after           {0}
seconds'.format(10-i),end='\r',flush=True)
                    time.sleep(1)
                print('\x1b[3A',end='\r')
            else:
                print('\033[1;32m Module dont ready,please put
the        antenna        outdoors        and        wait        a
moment\033[0m',end='\r',flush=True)
  except:
      x.close()
   exit()
```

Додаток Б

Уривок з історії отриманих шлюзом повідомлень під час проведення експерименту 30 травня 2024 року

receive message from node address with frequence 20,433.125MHz

message is b'99999.0,99999.0'

the packet rssi value: -82dBm

the current noise rssi value: -87dBm

receive message from node address with frequence 20,433.125MHz

message is b'87.0,84.0'

the packet rssi value: -68dBm

the current noise rssi value: -85dBm

receive message from node address with frequence 20,433.125MHz

message is b'90.0,63.0'

the packet rssi value: -49dBm

the current noise rssi value: -90dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.41442,30.618846667'

the packet rssi value: -61dBm

the current noise rssi value: -88dBm

receive message from node address with frequence 20,433.125MHz

message is b'33.0,21.0'

the packet rssi value: -50dBm

the current noise rssi value: -90dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.414011667,30.61893'

the packet rssi value: -53dBm

the current noise rssi value: -88dBm

receive message from node address with frequence 20,433.125MHz

message is b'17.0,17.0'

the packet rssi value: -65dBm

the current noise rssi value: -90dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.413735,30.618911667'

the packet rssi value: -71dBm

the current noise rssi value: -86dBm

receive message from node address with frequence 20,433.125MHz

message is b'17.0,16.0'

the packet rssi value: -68dBm

the current noise rssi value: -87dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.413465,30.618785'

the packet rssi value: -80dBm

the current noise rssi value: -89dBm

receive message from node address with frequence 20,433.125MHz

message is b'16.0,8.2'

the packet rssi value: -67dBm

the current noise rssi value: -87dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.413171667,30.618578333'

the packet rssi value: -73dBm

the current noise rssi value: -89dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.413171667,30.618578333'

the packet rssi value: -74dBm

the current noise rssi value: -88dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.41302,30.618183333'

the packet rssi value: -80dBm

the current noise rssi value: -88dBm

receive message from node address with frequence 20,433.125MHz

message is b'5.5,4.8'

the packet rssi value: -74dBm

the current noise rssi value: -88dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.412886667,30.61778'

the packet rssi value: -76dBm

the current noise rssi value: -81dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.412886667,30.61778'

the packet rssi value: -73dBm

the current noise rssi value: -86dBm

receive message from node address with frequence 20,433.125MHz

message is b'50.412781667,30.617295'

the packet rssi value: -82dBm

the current noise rssi value: -90dBm

receive message from node address with frequence 20,433.125MHz

message is b'6.1,3.5'

the packet rssi value: -80dBm

the current noise rssi value: -88dBm

Додаток В

Перелік записів обробленої бази даних з результатами експерименту

Кожен запис складається з 4 чисел, розділених між собою знаком ‘|’: перше число – номер запису, 2 число – широта, 3 число – довгота, 4 число – rssi. Запис під номером 151 має rssi рівне нулю, бо це запис з координатами шлюзу LoRa. Перелік записів:

1|50.406695|30.62153|-84
2|50.406755|30.621053333|-84
3|50.406781667|30.622|-83
4|50.406885|30.620655|-83
5|50.406891667|30.620766667|-84
6|50.406928333|30.620576667|-87
7|50.406995|30.62286|-89
8|50.407146667|30.623398333|-88
9|50.407166667|30.620165|-82
10|50.40763|30.625213333|-89
11|50.407895|30.626331667|-87
12|50.408005|30.619656667|-82
13|50.408008333|30.620185|-86
14|50.40802|30.621896667|-85
15|50.408378333|30.621813333|-86
16|50.40839|30.622603333|-87
17|50.408486667|30.618981667|-85
18|50.408743333|30.622303333|-86
19|50.408765|30.618658333|-87
20|50.40896|30.617506667|-87
21|50.408971667|30.622201667|-86
22|50.4091|30.622141667|-85
23|50.409103333|30.622071667|-85

24|50.409331667|30.622028333|-85

25|50.409361667|30.621878333|-87

26|50.409381667|30.616745|-88

27|50.40947|30.621976667|-86

28|50.40953|30.621893333|-85

29|50.409623333|30.621891667|-85

30|50.409656667|30.621818333|-86

31|50.409696667|30.621666667|-85

32|50.409713333|30.621821667|-76

33|50.409811667|30.6213|-85

34|50.409915|30.62205|-86

35|50.41002|30.620883333|-75

36|50.410021667|30.620988333|-83

37|50.410066667|30.616053333|-88

38|50.410096667|30.615648333|-89

39|50.410115|30.622286667|-86

40|50.410168333|30.62057|-81

41|50.410181667|30.615563333|-88

42|50.410251667|30.615656667|-88

43|50.41026|30.615733333|-88

44|50.41027|30.620435|-84

45|50.41036|30.616008333|-87

46|50.41054|30.620186667|-87

47|50.410556667|30.622326667|-85

48|50.410605|30.615981667|-88

49|50.410618333|30.615571667|-89

50|50.410735|30.619881667|-86

51|50.410751667|30.622136667|-85

52|50.410831667|30.615501667|-88

53|50.41091|30.619641667|-86

54|50.410981667|30.621976667|-85

55|50.41102|30.615226667|-86

56|50.411035|30.615211667|-85

57|50.411128333|30.619425|-84

58|50.411151667|30.615426667|-87

59|50.411243333|30.621851667|-87

60|50.41133|30.619168333|-85

61|50.411348333|30.62183|-86

62|50.411443333|30.619228333|-87

63|50.411521667|30.621756667|-85

64|50.411676667|30.615545|-83

65|50.411813333|30.618756667|-88

66|50.411815|30.621753333|-77

67|50.411875|30.627361667|-85

68|50.411915|30.621665|-84

69|50.412003333|30.621558333|-84

70|50.412048333|30.617178333|-81

71|50.412058333|30.618715|-87

72|50.412093333|30.616175|-81

73|50.412171667|30.627141667|-84

74|50.412173333|30.617711667|-75

75|50.412206667|30.617368333|-87

76|50.412206667|30.618135|-74

77|50.41224|30.617303333|-80

78|50.412241667|30.618555|-73

79|50.412265|30.62152|-83

80|50.412298333|30.616585|-72

81|50.412338333|30.61731|-79

82|50.412345|30.617303333|-80

83|50.412441667|30.617311667|-79

84|50.412476667|30.61726|-79

85|50.412545|30.621493333|-85

86|50.412556667|30.617308333|-82

87|50.412781667|30.617295|-82

88|50.412793333|30.617361667|-85

89|50.412815|30.621498333|-76

90|50.412831667|30.615661667|-86

91|50.412886667|30.61778|-76

92|50.412888333|30.626843333|-83

93|50.412903333|30.617745|-78

94|50.412913333|30.617895|-85

95|50.412991667|30.625271667|-83

96|50.413|30.624886667|-84

97|50.413016667|30.618146667|-78

98|50.41302|30.618183333|-80

99|50.413025|30.623701667|-81

100|50.41303|30.621776667|-83

101|50.413036667|30.624358333|-77

102|50.41306|30.624055|-83

103|50.413068333|30.624411667|-84

104|50.413073333|30.62215|-76

105|50.413075|30.623395|-84

106|50.413163333|30.62305|-84

107|50.413168333|30.618511667|-88

108|50.413171667|30.618578333|-73

109|50.413198333|30.62666|-82

110|50.41321|30.624063333|-85

111|50.413225|30.6134|-86

112|50.413226667|30.62475|-79

113|50.413241667|30.622533333|-81

114|50.413326667|30.623615|-84

115|50.413338333|30.622711667|-84

116|50.413345|30.622818333|-84

117|50.413363333|30.623183333|-82

118|50.413373333|30.618763333|-68

119|50.4134|30.624865|-81

120|50.413465|30.618785|-80

121|50.41348|30.626403333|-82

122|50.413491667|30.622445|-77

123|50.41353|30.624758333|-84

124|50.41353|30.624778333|-77

125|50.41358|30.618748333|-75

126|50.413581667|30.615366667|-81

127|50.413656667|30.613296667|-86

128|50.413696667|30.615485|-78

129|50.41372|30.622095|-83

130|50.413735|30.618911667|-71

131|50.41376|30.615845|-83

132|50.413761667|30.62462|-82

133|50.413831667|30.618718333|-67

134|50.413843333|30.616283333|-80

135|50.413865|30.6162|-83

136|50.413976667|30.616515|-80

137|50.414003333|30.616548333|-79

138|50.414008333|30.613746667|-86

139|50.414011667|30.61893|-53

140|50.414021667|30.626091667|-82

141|50.414043333|30.616595|-78

142|50.414075|30.61663|-79

143|50.414091667|30.618763333|-53

144|50.414165|30.616828333|-80

145|50.41419|30.616806667|-73

146|50.414213333|30.618475|-65

147|50.414215|30.618496667|-71

148|50.414248333|30.61854|-71

149|50.414293333|30.617313333|-70

150|50.414323333|30.617161667|-74

151|50.41438|30.61855|0

152|50.41442|30.618846667|-61

153|50.414428333|30.618278333|-59

154|50.414455|30.614073333|-86

155|50.4145|30.61826|-64

156|50.414525|30.617573333|-70

157|50.414526667|30.618276667|-63

158|50.414565|30.61788|-61

159|50.414575|30.618078333|-59

160|50.414601667|30.62567|-82

161|50.414611667|30.617715|-82

162|50.414621667|30.62443|-82

163|50.414653333|30.618388333|-69

164|50.414655|30.621083333|-85

165|50.41466|30.6246|-82

166|50.414671667|30.624158333|-82

167|50.414678333|30.618388333|-74

168|50.414718333|30.618705|-60

169|50.414718333|30.621428333|-81

170|50.41472|30.618315|-60

171|50.414743333|30.618278333|-62

172|50.41475|30.620696667|-77

173|50.41477|30.6185|-71

174|50.414795|30.625055|-81
175|50.414813333|30.618618333|-65
176|50.414848333|30.618478333|-63
177|50.41486|30.625448333|-81
178|50.414861667|30.618461667|-64
179|50.414898333|30.618635|-61
180|50.414901667|30.618416667|-63
181|50.41494|30.618441667|-63
182|50.414943333|30.625413333|-82
183|50.414981667|30.618463333|-63
184|50.414986667|30.620093333|-81
185|50.415025|30.614421667|-86
186|50.415085|30.618556667|-63
187|50.415151667|30.62156|-82
188|50.41518|30.615253333|-86
189|50.415206667|30.61886|-70
190|50.415246667|30.618776667|-68
191|50.41534|30.618908333|-73
192|50.41535|30.62126|-83
193|50.415366667|30.618318333|-64
194|50.415511667|30.616091667|-85
195|50.415526667|30.618906667|-73
196|50.415571667|30.618621667|-66
197|50.415571667|30.618758333|-67
198|50.415585|30.61603|-86
199|50.415628333|30.618953333|-82
200|50.41578|30.61589|-83
201|50.415826667|30.61901|-83
202|50.415925|30.615726667|-85
203|50.415936667|30.616578333|-82

204|50.415976667|30.615723333|-86

205|50.416031667|30.616933333|-84

206|50.416166667|30.617335|-85

207|50.416213333|30.618531667|-84

208|50.41622|30.615686667|-85

209|50.416258333|30.619021667|-80

210|50.416316667|30.615803333|-82

211|50.416375|30.617728333|-86

212|50.41649|30.615786667|-85

213|50.416561667|30.615876667|-84

214|50.416648333|30.615808333|-84

215|50.416715|30.615783333|-86

216|50.417126667|30.61609|-83

217|50.4172|30.616046667|-86

218|50.417356667|30.615968333|-85

219|50.417388333|30.616166667|-85

220|50.417578333|30.615801667|-83

221|50.417628333|30.615948333|-81

222|50.417793333|30.618946667|-84

223|50.417796667|30.615605|-85

224|50.417853333|30.61573|-85

225|50.418016667|30.615363333|-85

226|50.41809|30.615463333|-86

227|50.418316667|30.615168333|-86

228|50.418328333|30.615266667|-86

229|50.418568333|30.615178333|-86

230|50.41859|30.615076667|-86

231|50.418656667|30.615183333|-85

232|50.418913333|30.615093333|-86

233|50.41906|30.61485|-85

234|50.41915|30.614908333|-85

235|50.419338333|30.61475|-85

236|50.419375|30.614755|-85

237|50.419505|30.612163333|-82

238|50.41951|30.612163333|-85

239|50.419561667|30.614536667|-85

240|50.419565|30.612328333|-86

241|50.41958|30.614543333|-79

242|50.419608333|30.612246667|-83

243|50.419726667|30.614325|-84

244|50.41978|30.612351667|-75

245|50.419783333|30.614333333|-85

246|50.419791667|30.612385|-82

247|50.419868333|30.612706667|-83

248|50.4199|30.612711667|-85

249|50.419928333|30.613073333|-86

250|50.419933333|30.61411|-86

251|50.420011667|30.614198333|-86

252|50.420028333|30.613141667|-85

253|50.420095|30.613365|-85

254|50.420103333|30.613513333|-86

255|50.420168333|30.614025|-86

256|50.420205|30.613821667|-86

257|50.420236667|30.614035|-84

258|50.41x56<s33|70.615178333|-86

259|50.446683333|30.458123333|-28

260|50.44669|30.458126667|-27