КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інформаційних систем та технологій

Спеціальність 126 – Інформаційні системи та технології
Освітня програма «Програмні технології інтернет речей»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

«Створення IoT системи моніторингу сейсмологічної ситуації на прикладі одного з регіонів України»

Здобувач освіти на освітньому рівні «Магістр» 2-го курсу групи ІРма-21:

Дмитро САХАРОВ

(Власне Ім'я, ПРІЗВИЩЕ)

(підпис студента)

Науковий керівник:

к.т.н., доцент

(науковий ступінь, вчене звання)

Ольга КРАВЧЕНКО

(Власне Ім'я, ПРІЗВИЩЕ)

(дата) (підпис)

Попередній захист:

(Висновок: "До захисту в Екзаменаційній комісії")

В.о. завідувача кафедри

Інформаційних систем та технологій

д.т.н., доцент

(науковий ступінь, вчене звання)

Олексій КОЛЕСНІКОВ

(підпис) (Власне Ім'я, ПРІЗВИЩЕ) (дата)

Київ 2021

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційні системи та технології
Освітній рівень Магістр
Спеціальність 126 Інформаційні системи та технології
Освітня програма Програмні технології інтернет речей

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри,
д.т.н., доцент
Олексій КОЛЕСНІКОВ

_____
«__» _____ 2021 року

ЗАВДАННЯ

НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА

Здобувач освіти: Дмитро САХАРОВ

Група: ІРма-21

1. **Тема кваліфікаційна робота магістра**: «Створення IoT системи моніторингу сейсмологічної ситуації на прикладі одного з регіонів України».

Затверджена протоколом засідання кафедри ІСТ №16/20 від 09 листопада 2020р.

2. **Строк подання студентом готової роботи** – «20» травня 2021 р.

3. **Цільова установка та вихідні дані до роботи**: дослідження в області систем раннього попередження про стихійні лиха. Програмно-апаратні рішення для побудови системи моніторингу сейсмологічної ситуації з використанням систем IoT.

4. **Зміст роботи:** Розділ 1 Поняття систем раннього попередження та пошук аналогів (аналіз аналогів, вхідні дані для вирішення задачі кваліфікаційної роботи); Розділ 2 Розробка проекту IoT рішення та методів обробки даних в IoT (Розробка проекту IoT рішення, методи та засоби детекції землетрусів, знаходження епіцентру землетрусів, використання хмарних технологій для реалізації IoT рішень); Розділ 3 Реалізація архітектури IoT рішення (архітектура проекту IoT рішення, комунікаційні технології та системи, схемотехнічне проектування); Розділ 4 Програмна реалізація інтерфейсних рішень, програмування мікроконтролерів та реалізація проекту,

використання розподілених систем та хмарних технологій (програмування мікроконтролера STM32, розробка програмного забезпечення та графічного інтерфейсу, аналіз результатів досліджень).

5. **Перелік графічного матеріалу**: Концептуальна модель архітектури системи IoT для моніторингу сейсмологічної ситуації; алгоритм роботи системи та структурна схема; блок схема роботи програмного забезпечення; графічний інтерфейс системи для ОС Windows.

6. **Календарний план виконання роботи:**

Етапи виконання дипломних робіт	**Термін виконання**	**Результат виконання**
1. Вибір тематики кваліфікаційної роботи магістра	20.10.2019	виконано
2. Наказ про затвердження тем кваліфікаційної роботи магістра та призначення наукових керівників	09.11.2020	виконано
3. Формування переліку нормативних матеріалів, літератури з проблематики кваліфікаційної роботи магістра роботи	12.12.2020	виконано
4. Розробка плану кваліфікаційної роботи магістра і його погодження з науковим керівником	26.12.2020	виконано
5. Написання I розділу дипломної роботи	29.01.2021	виконано
6. Написання II розділу дипломної роботи	26.02.2021	виконано
7. Написання III розділу дипломної роботи	09.03.2021	виконано
8. Написання IV розділу дипломної роботи	26.03.2021	виконано
9. Підготовка висновків і пропозицій	07.04.2021	виконано

9. Попередній захист дипломної роботи	20.04.2021	виконано
10. Рецензування кваліфікаційної роботи магістра	до 20.05.2021	виконано
11. Захист кваліфікаційної роботи магістра	26.05.2021	

Дата видачі завдання «_10_» _листопада_ 2020 р.

Керівник роботи: к.т.н., доц. Ольга КРАВЧЕНКО _____ (підпис)
Завдання прийняв до виконання:
Здобувач освіти на освітньому рівні «Магістр» 2-го курсу групи ІРма-21

Дмитро САХАРОВ _____
(Власне Ім'я, ПРІЗВИЩЕ) (підпис)

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційних систем та технологій

Освітня програма «Програмні технології інтернет речей»

Кваліфікаційна робота магістра Дмитра САХАРОВА

Тема роботи: «Створення IoT системи моніторингу сейсмологічної ситуації на прикладі одного з регіонів України».

Мета кваліфікаційної роботи магістра – створення IoT системи, що складалася б з сейсмографів, що здатні реєструвати землетруси, та передавати на центр обробки і аналізу, що визначав би теоретичний епіцентр землетрусу. Центр обробки і аналізу повинен інформувати жителів регіону про небезпеку.

Об'єкт дослідження – IoT екосистема для моніторингу сейсмологічної ситуації на прикладі Закарпатського регіону.

Предмет дослідження – спосіб реєстрації землетрусів, визначення епіцентру коливань, розробка відповідного апаратного та програмного забезпечення для отримання та обробки даних.

Методи дослідження – використано теоретичні та емпіричні методи дослідження: аналіз та узагальнення наукової літератури, визначення вхідних даних, далі – розробка програмного коду для мікроконтролера STM32, проектування хмарного програмного забезпечення та графічного інтерфейсу користувача, додатку для смартфону для попередження населення.

Практичне значення одержаних результатів. Розроблений прототип системи IoT рішень для моніторингу сейсмічної активності у Закарпатському регіоні, може здійснювати сповіщення населення даного регіону про землетрус, тим самим запобігаючи людським жертвам та значним руйнування інфраструктури.

Апробація результатів. Основні положення і результати досліджень, викладені у проекті, пройшли апробацію на VII інтернаціональній конференції «Information Technology and Interactions» (Satellite) (Київ, 2020) [1], на шістнадцятій міжнародній науково-технічній конференції «Проблеми інформатизації» у 2021 році [2] та на шостій міжнародній науково-практичній конференції «Інформаційні технології та взаємодії» (Київ, 2019) [3].

Кваліфікаційна робота магістра складається зі змісту, вступу, основної частини, яка включає чотири розділи, висновків та списку використаних джерел. Всього _____ сторінок.

КЛЮЧОВІ СЛОВА: архітектура IoT рішення, мікроконтроллер, апаратний комплекс, програмний комплекс, схемотехнічне рішення, сейсмограф, сейсмічність.

ABSTRACT

TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of Information Technologies

Department of Information Systems and Technologies

Educational Program "Software Technologies of the Internet of Things"

Qualification work of master Dmytro SAKHAROV.

Work topic: " IoT system of seismological situation monitoring ".

The purpose of the master's qualification work is to develop an IoT system for seismological situation monitoring in one of the regions of Ukraine. The system should have seismic sensors and a control center. The system should warn residents of the region about possible earthquake hazard.

The object of research is an IoT system for seismological situation monitoring in the Zakarpatska oblast.

The subject of research is the solution for earthquake detection, finding the epicenter location. Development of the hardware and software for data acquisition and processing.

Research methods. Used theoretical and empirical research methods: analysis and generalization of scientific literature, requirement agregation. Those lead to development of software for STM32 microcontroller, software design of graphical user interface, developed an application MVP to warn residents about hazard.

The practical significance of the obtained results. This qualification work of the master has an applied value - the development of an IoT system for seismological situation monitoring in one of the regions of Ukraine. This system could warn residents about the earthquake hazard. An early warning could prevent human victims and major infrastructure destruction.

Approbation of results. The main provisions and research results presented in the project were tested at the VII International Conference "Information Technology and Interactions" (Satellite) (Kyiv, 2020) [1] and at the sixteenth international scientific and technical conference "Problems of Informatization" in 2021 [2], and at the sixth International Conference. Information Technology and Interactions" (Kyiv, 2019) [3]

The master's qualification work consists of the content, introduction, main part, which includes four sections, conclusions and a list of sources used. Total _____ pages.

KEY WORDS: IoT solution architecture, microcontroller, seismic sensor, earthquake

ЗМІСТ

ВСТУП

За останні кілька років IoT став однією з найважливіших технологій 21 століття. Тепер, коли ми можемо під'єднати побутові предмети – кухонні прилади, машини, термостати до Інтернету через вбудовані пристрої, можливе безперебійне спілкування між людьми, речами(пристроями).

За допомогою недорогих хмарних обчислень, великих даних, аналітики та мобільних технологій фізичні речі можуть обмінюватися та збирати дані з мінімальним втручанням людини. У цьому гіперпов'язаному світі цифрові системи можуть записувати, контролювати та регулювати кожну взаємодію між підключеними речами. Фізичний світ відповідає цифровому світу, і вони співпрацюють для комфорту людини [4].

Землетрус – це випадкове стихійне лихо, яке завдає шкоди життю та майну. Ми не можемо зупинити чи протидіяти землетрусам, проте ми можемо бути своєчасно повідомленими про них. В даний час є численні досягнення, які можна використовувати для виявлення маленьких трясок і вібрацій землі, для того, щоб вжити запобіжних заходів, перш ніж відбудуться основні коливання всередині землі. Життя можна врятувати, якщо завчасно виявити землетрус.

Значні сейсмічні процеси існують на заході України, зокрема на Закарпатті, де неодноразово були землетруси інтенсивністю 6-7 балів. Епіцентри цих землетрусів знаходилися в районах Свалява, Довгого, Теребасви, Мукачевого, Ужгорода [5]. Тому прикладом регіону для створення системи моніторингу сейсмологічної активності може слугувати Закарпатська область.

У рамках проектування було проаналізовано існуючі рішення та алгоритми, що дозволяють покращити точність детектування землетрусів.

Щоб мати змогу передавати дані про землетруси одразу зі всіх сейсмографів у регіоні, протокол передачі даних має забезпечувати безперебійну доставку пакетів, а сервер що слугує центром обробки і аналізу має бути здатним приймати декілька десятків потоків даних одночасно. Задача полягає у розробці пристрою сейсмографу, серверного додатку для прийому даних з сейсмографу.

Додаток має проектуватися з урахуванням майбутнього масштабування екосистеми до сотні датчиків-сейсмографів.

Новизною даного проекту є використання новітньої елементної бази, зокрема, стільникового модему для забезпечення 2G/3G з'єднання, та проектування і використання алгоритмів для безперебійної передачі даних у високонавантажених системах.

Мета роботи – створення IOT системи, що складалася б з сейсмографів, що здатні реєструвати землетруси, та передавати на центр обробки і аналізу, що визначав би теоретичний епіцентр землетрусу. Центр обробки і аналізу повинен інформувати жителів регіону про небезпеку.

Для досягнення мети необхідно виконати наступні завдання:

- провести аналіз існуючих систем інформування про стихійні лиха;
- виконати дослідження технології зв'язку для даної системи;
- сформувати уявлення про конструктивні та структурні рішення;
- описати логічні зв'язки вузлів системи на етапі проектування;
- навести структурну схему роботи системи IoT для моніторингу та інформування про землетруси;
- виконати верифікацію даних.

Об'єктом дослідження є IOT екосистема для моніторингу сейсмологічної ситуації на прикладі Закарпатського регіону. Апаратно-програмний комплекс належить до інформаційних систем попередження про стихійні лиха.

Предметом дослідження є спосіб реєстрації землетрусів, визначення епіцентру коливань, розробка відповідного апаратного та програмного забезпечення для отримання та обробки даних.

Практичною цінністю проекту є розробка прототипу системи IoT рішень для моніторингу сейсмічної активності у Закарпатському регіоні, сповіщення населення про землетрус, тим самим запобігаючи людським жертвам та значним руйнування інфраструктури.

Апробація результатів. Основні положення і результати досліджень, викладені у проекті, пройшли апробацію на VII інтернаціональній конференції «Information Technology and Interactions» (Satellite) (Київ, 2020) [1], на шістнадцятій міжнародній науково-технічній конференції «Проблеми інформатизації» у 2021 році [2] та на шостій міжнародній науково-практичній конференції «Інформаційні технології та взаємодії» (Київ, 2019) [3].

Обсяг і структура кваліфікаційної роботи магістра. Робота складається зі вступу, 4 розділів, висновку та 55 використаних інформаційних джерел.

Розділ 1. Поняття систем раннього попередження та пошук аналогів

1.1 Поняття систем раннього попередження

Системи моніторингу та раннього попередження – це комбінація інструментів та процесів, закладених в інституційні структури, координовані державними чи міжнародними регуляторами. Незалежно від того, чи вони зосереджені на одній конкретній природній небезпеці чи на багатьох, ці системи складаються з чотирьох елементів: знання про ризик, технічний моніторинг та служба попередження, поширення попереджень про небезпеку людям, що ризикують постраждати, та обізнаності громадськості і готовність протидіяти можливим майбутнім наслідкам. Служби що набувають знання про ризик та процеси попередження населення лежать в основі цих систем, а від того, наскільки добре вони функціонують, залежить яку кількість жертв вдалось попередити, та на скільки вдалось зменшити суму збитків. Науково-технічний прогрес сприяв значному покращенню якості, своєчасності та часу попередження про небезпеку. Покращення апаратних та програмних технологій дає можливість використовувати смартфони та Інтернет речей для моніторингу навколишнього середовища в режимі реального часу.

Перші міжнародно визнані зусилля щодо розвитку СРП були досягнуті кількома конференціями по даній тематиці. У 1998 році - перша в історії міжнародна конференція з питань раннього попередження (EWC’98) відбулась у Потсдамі, Німеччина. Тут міжнародна спільнота зібралася щоб обговорити сучасні знання та підходи щодо СРП. П'ять років потому, у 2003 році, у Бонні відбулася друга конференція (EWC II). На цій конференції увага була зосереджена на тому, як інтегрувати функцію раннього попередження у відповідну державну політику. На конференції підкреслили, що, хоча ранні попередження насамперед ґрунтуються на технічній інформації та моніторингу ризиків, політика громадськості повинна гарантувати що у відповідь на попередження, слідує відповідна реакція. Дискусія, ініційована конференцією була фундаментальною для майбутнього розвитку СРП, оскільки вона надала

повноваження місцевим органам влади, місцевим установам, та громади брати участь у цілому процесі формування політик реакції на стихійні лиха, сприяючи обізнаності та готовності до реагування на загрози. Це означало перехід для СРП із суто технічного продукту до соціально-політичного процесу.

1.1.1 Чотири складові системи раннього попередження.

Раннє попередження – це стратегія, прийнята багатьма суспільствами та спрямована на зменшення наслідків стихійних лих. СРП часто базуються на взаємозв'язку між візуальними спостереженнями, минулим досвідом та співпрацею для зменшення втрат від майбутніх небезпек. Стратегії раннього попередження – це комплекс заходів прийнятих для підвищення стійкості. При правильній реалізації СРП може допомогти зменшити втрати життя та майна, а також мінімізувати екологічну шкоду. Все це поєднується у вигідному співвідношенні витрат на впровадження СРП, та вигоди від попередженних матеріальних збитків та підвищення безпеки. Для ефективної системи раннього попередження, необхідна взаємодія чотирьох ключових елементів (рис. 1.1): знання ризиків (risk knowledge), служби моніторингу та попередження (monitoring and warning service), розповсюдження та можливість спілкування (dissemination and communication) та здатність реагувати (response capability).

Рисунок 1.1 – Ключові елементи систем раннього попередження [6]

Служби моніторингу та попередження це найчастіше те, що спадає на думку, говорячи про СРП. Це інфраструктура, яка забезпечує прогнози та попередження. Моніторинг є процесом збору інформації, про чинники та явища, пов'язані з ризиком, такі як дощ (що спричиняє повінь, або засуху за його відсутності), або сейсмічні хвилі (співвідносні із землетрусами та руйнацією інфраструктури). Це можна зробити за допомогою прямих спостережень, наприклад побачивши наближення пожежі або зсуву грунту. Однак такі спостереження мають обмежену корисність через близькість загрози та обмежені можливості прийняття дії щодо зменшення ризику. Подібна проблема виникає при землетрусах, де сейсмометри здатні викликати попередження за кілька секунд до удару, тому час реакції усієї системи, від перших поштовхів, до сповіщення на смартфон, є дуже критичним. Сучасні технології надають можливість агрегації та обробки даних з декількох джерел моніторингу та на високій швидкості, створення можливостей для підвищення точності та швидкості, з якою можна складати прогнози стихійних лих.

1.2 Вивчення аналогів

1.2.1 Earthquake early warning system by IOT using Wireless sensor networks

У статті авторів Alphonsa A. та Ravi G., наведено рішення, використовуючи технологію WSN [7]. Технологія базується на розподілені сенсорів на поверхні, що аналізується, так що вони формують сітку. Кожний сенсор це PIC мікроконтролер, з трансміттером та трьома акселерометрами, як датчики вібрації. Сенсор з’єднується зі шлюзом, використовуючи стандарт ZigBee. Шлюзом є персональний комп’ютер що по послідовному порту з’єднується с ZigBee трансміттером. Висновок: не зважаючи на те, що ZigBee підтримує режим mesh network, завдяки якій кожний трасміттер може бути підключений до шлюзу, використовуючи проміжні пристрої, зона дії всеодно дуже маленька – кількасот метрів на прямій видимості. Впевнений, що чутливості акселерометрів достатньо щоб охопити декілька десятків кілометрів в плані детектування

вібрацій. Але, використовуючи рішення, що наведено у статті, існує необхідність розміщувати сенсори кожні кількасот метрів, що є економічно невигідним. Сейсмічна хвиля мандрує так швидко (2-8 км/с), що така висока щільність розподілення сейсмографів взагалі ні до чого. Також, навантаження на трансміттери, що знаходяться найближче до шлюзу буде дуже велике, і, у критичні моменти, коли сейсмічна хвиля буде розповсюджуватись, і трафік з критичними даними буде дуже об'ємний, це може значно обмежити швидкість отримання шлюзом необхідної інформації. Тобто, дорогоцінний час на реагування після сповіщення про землетрус буде зменшено.

1.2.2 A Smart IoT Device for Detecting and Responding to Earthquakes

Інший аналог, описаний у статті «A Smart IoT Device for Detecting and Responding to Earthquakes», використовує технологію Wi-Fi для підключення сенсорів [8]. Розроблена система також складається з 32-бітового процесора (ARM Cortex-M3), bluetooth, зумера. Програмне забезпечення розробленого пристрою виявлення землетрусів відіграє ключову роль у виявленні землетрусів та передачі тривожного повідомлення на сусідні смарт-пристрої, такі як смартфон, смарт-годинник, AI-динамік, пристрої домашньої автоматизації тощо. Для цього система приймає дані про прискорення як вхідні та запускає алгоритм виявлення землетрусу під час спостереження рухів вище певного рівня прискорення (0,02 g). Оскільки виявлений сигнал може бути землетрусом або помилковим сигналом тривоги, алгоритм виявлення класифікує вхідний сигнал з двосекундним вікном на предмет чи це землетрус, чи ні. Якщо результат алгоритму виявлення землетрусу перевищує поріг (тобто показник подібності 95%) протягом 10 секунд, алгоритм переходить у другу фазу виявлення землетрусу, де алгоритм виявлення вивчає наступні три секунди даних прискорення і підтверджує його як землетрус. Щоб зменшити ймовірність помилкових тривог, у роботі використовується двофазний алгоритм виявлення землетрусів. Потім пристрій виявлення землетрусу надсилає попереджувальне повідомлення на прилеглі пристрої через Bluetooth low energy з рівнем коливання

(тобто рівнем інтенсивності) шляхом перетворення вимірюваного значення пікового наземного прискорення у три рівні коливань. У той же час пристрій попереджає користувачів, які знаходяться поблизу, за допомогою внутрішнього зумера та світлодіодного світла. Схема алгоритму виявлення землетрусів наведена на рис. 1.2.

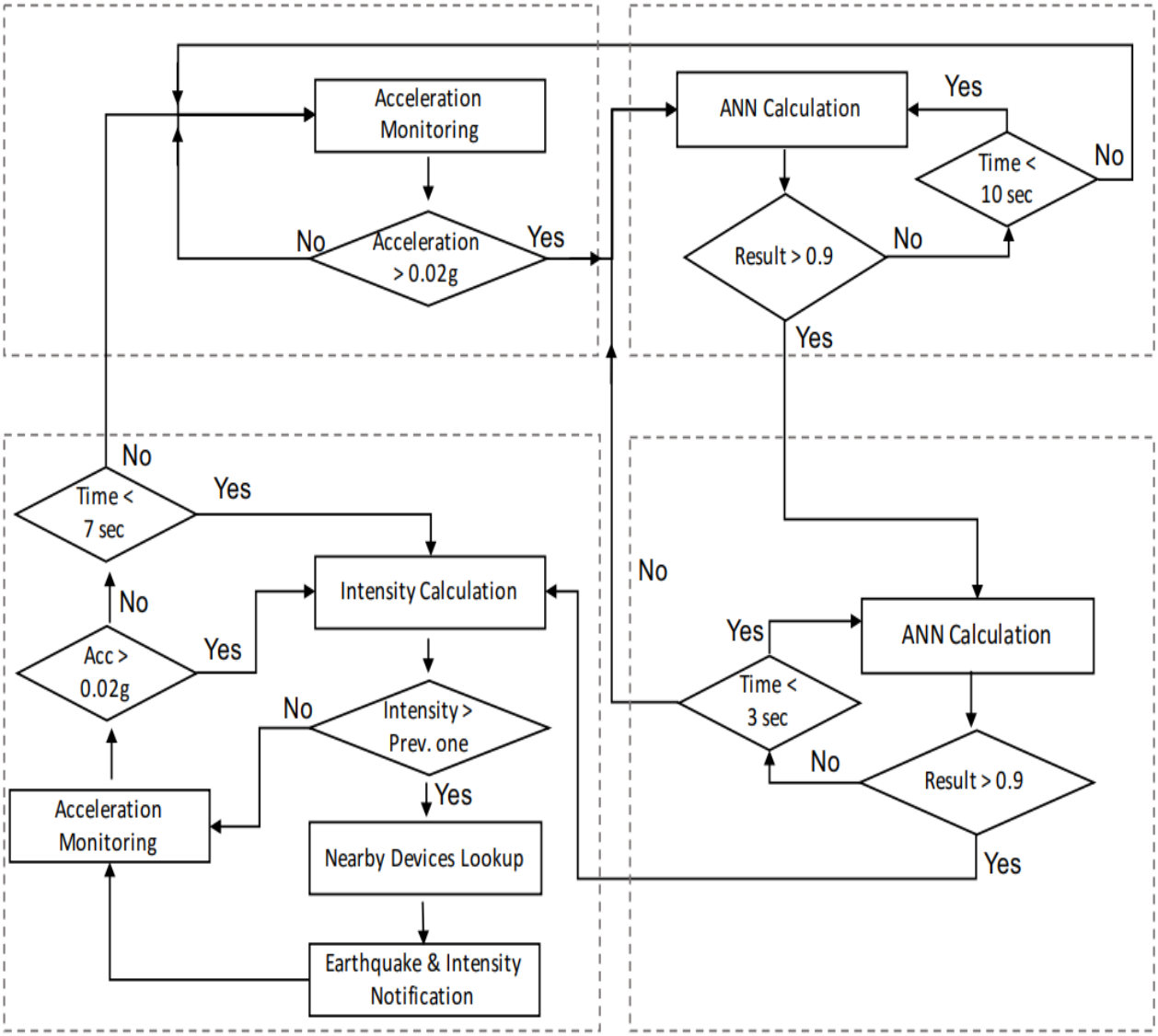

Рисунок 1.2 – Схема алгоритму детектування землетрусів [8]

У статті детально описується алгоритм виявлення землетрусів за допомогою штучної нейронної мережі (Artificial Neural Network) [9], за допомогою простої методики машинного навчання, яка широко застосовується протягом останніх декількох десятиліть. Модель ANN має три нейрони у

вхідному шарі, п’ять нейронів у прихованому шарі та один нейрон у вихідному шарі, як показано на рис. 1.3.

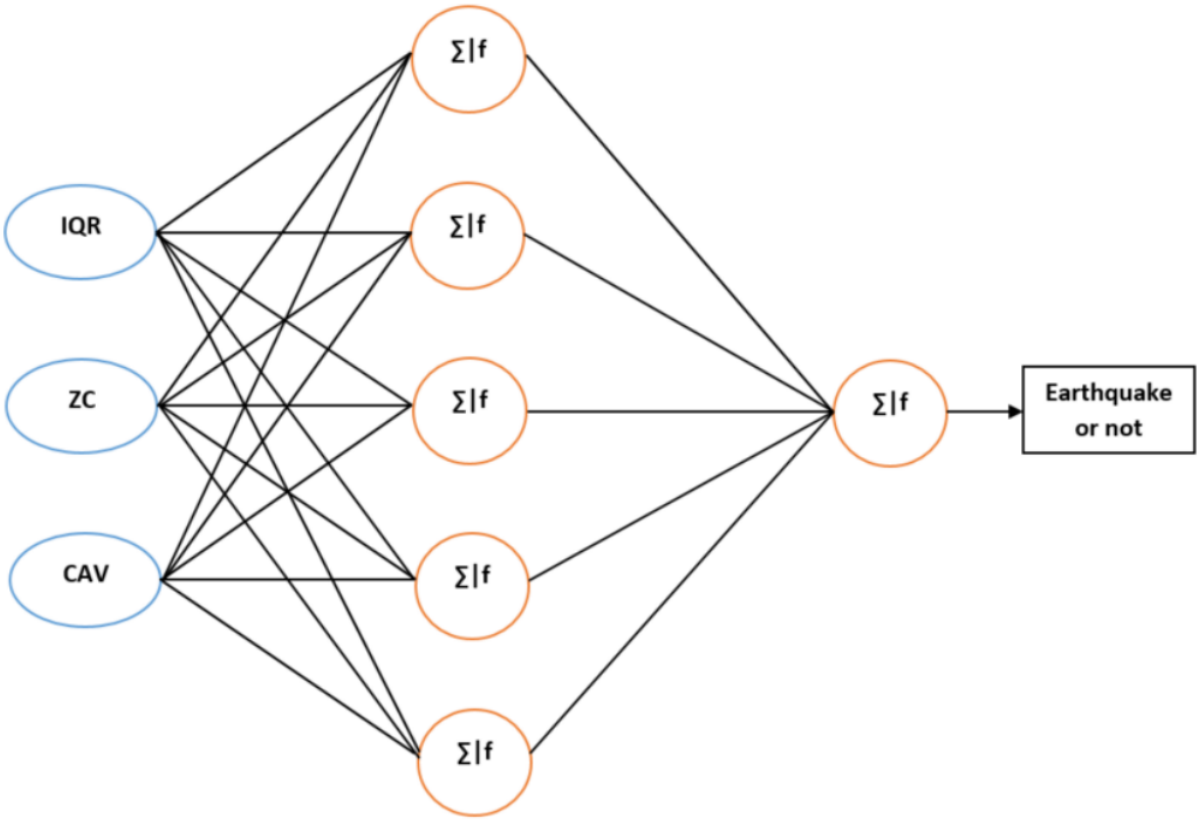

Рисунок 1.3 – Нейронна мережа для детектування [8]

Оскільки ефективність моделі машинного навчання сильно залежить від набору даних, що використовуються для навчання та тестування, дані потрібно попередньо обробити перед навчанням моделі. З цією метою розробники у якості попередньої обробки балансували та масштабували набори даних. Оскільки на фазі навчання використовувались як дані про землетруси, так і дані про не землетруси, які мають різний розмір даних, потрібно збалансувати набір даних між даними про землетруси та не землетрусами. Щоб збалансувати набір даних, використано алгоритм кластеризації K-середнього.Розроблений пристрій сповіщення про землетрус може надсилати попереджувальне повідомлення на сусідні пристрої, такі як смартфон, смарт-годинник, AI-динамік, пристрої домашньої автоматизації, використовуючи Bluetooth або Wi-Fi. Щоб реалізувати

цей сценарій, було розроблено два випадки використання. Перший – розумні пристрої У цьому випадку пристрій сповіщення про землетрус надсилає попереджувальне повідомлення на сусідні пристрої, такі як смартфон чи телевізор, що виконує відповідні екстрені команди. На рис. 1.4 показані ілюстрації, що відображаються на смартфоні чи телевізорі, та приклад ситуації.

Рисунок 1.4 – Випадок використання з розумними пристроями [8]

Автоматизація будинку: У цьому сценарії, відповідно до рівня коливань, підключені пристрої домашньої автоматизації, такі як електроенергія, газ та водопровідна вода, автоматично відключаються, як описано на рис. 1.5.

Рисунок 1.5 – Інтеграція у домашню автоматику

Отже, в статті добре описані алгоритми детектування землетрусу, використовуючи нейронну мережу, що була навчена на існуючих даних про землетруси. Також багато уваги приділено описанню інформаційного середовища для інформування населення. Проте, використання Wi-Fi робить пристрій залежним від розміщення точки доступу неподалік сейсмографу. Або, можна, сказати, це рішення для «домашньої» автоматики.

1.2.3 Technologies of Internet of Things applied to an Earthquake Early Warning System

У третьому проаналізованому проекті «Technologies of Internet of Things applied to an Earthquake Early Warning System», автори використовували дані про землетруси в Еквадорі – країні, в якій у якій всередньому відбувається 6 землетрусів в день (дані 2013-го року) [10]. Дослідники стверджують, що завдяки використанню MQTT – надлегкого протоколу, вдалось досягти передбачення максимального піку за 12 секунд поблизу зони епіцентру. Сіткою сенсорів виступають смартфони, а точніше акселерометри всередині них. У роботі використовуються часовий та просторовий аналіз, які не присутні в інших подібних роботах, роблячи систему більш адаптивною для різних географічних зон та наявних ресурсів. Так як команда дослідників розробила додаток для смартфону, у роботі також проведено порівняльний аналіз показників енергоспоживання та використання пам’яті додатком.

Мережа акселерографів, що представлена в цій роботі, базується на тришаровій ієрархії та архітектурі EEWS, як показано на рис. 1.6.

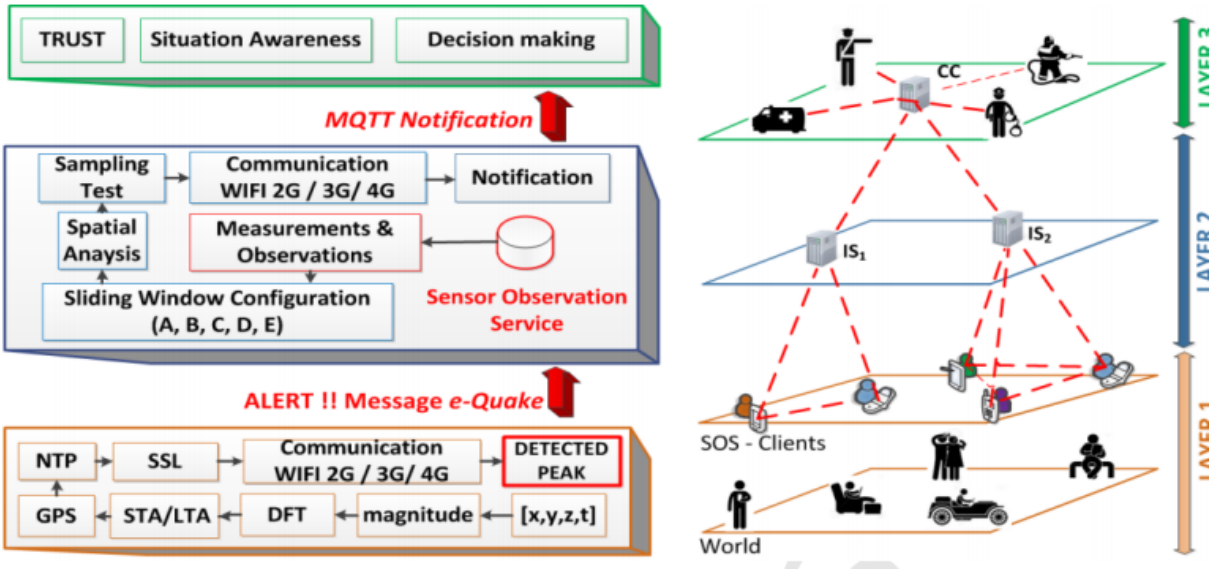

Рисунок 1.6 – Тришарова архітектура проекту; Рівень 1: Сенсорна мережа; Рівень 2: проміжний сервер; Шар 3: Центр управління

У першому шарі смартфон використовується як одиниця обробки та відправляє зразки до одного з проміжних серверів, який відповідає другому рівню системи, як тільки смартфон виявляє сейсмічний пік, за допомогою математичного алгоритму, який був спеціально розроблений для виконання вимог системи реального часу з точки зору швидкодії. Кожен проміжний сервер вирішує, сталася сейсмічна подія чи ні, і негайно повідомляє користувачів які територіально відносяться до данного серверу, в той же час, повідомляє про інцидент контрольний центр, що є третім і останнім шаром. Контрольний центр агрегує в собі додатки та алгоритми, які приймають рішення на основі інформації, що прибуває з проміжних серверів.

Рівень 1 – клієнтський додаток та процесинг прискорення. Додаток для смартфону повинен бути простим, не втручатися в щоденні дії користувача та не збільшувати споживання батареї. Акселерограма визначається як об'єднання сейсмічного сигналу і шуму проти часу. Використовується дискретне перетворення Фур'є (DFT), для переходу від часової на частотну область, що дозволяє застосовувати фільтри низьких частот для видалення високих частот, що відповідають шуму, який безпосередньо впливає на піки сигналу. DFT

виконується за алгоритмом швидкого перетворення Фур'є (ШПФ) для визначенних часових вікон, так як алгоритм неможливо застосувати з нескінченною часовою областю. Далі спрацьовує алгоритм короткострокового та довгострокового усереднення (STA / LTA) [11]. Він використовується через його широкі можливості виявлення подій у сейсмології, низької кількості обчислень, та, внаслідок цього – низької енергії споживання. Тоді як STA дозволяє розрахувати теперішнє значення (VA) з останніх N семплів, LTA дозволяє наблизити передбачуване значення (VP) з M семплів, де N <M, тому семпли STA входять у семпли LTA. А сейсмічним піком вважається, якщо співвідношення VA та VP перевищує певний поріг. Цей поріг є динамічним, щоб відрізняти періодичні рухи користувача (біг або ходьба) і справжній сейсмічний пік. Цей поріг збільшується, якщо відношення VA до VP збільшується, в іншому випадку поріг зменшується. Зразки з меншим значенням, ніж розраховане порогове значення, відкидаються. Потім програма отримує доступ до GPS, щоб отримати дані користувача про поточне місцезнаходження, що необхідно для процесингу на проміжному сервері. У режимі реального часу, системі необхідно підтримувати однакове поняття часу у всій архітектурі, для цього протокол NTP реалізований для синхронізації всередині всієї архітектури.

Рівень 2 – проміжний сервер, який забезпечує надійність системи з наступними припущеннями:

- семпли першого шару не залежать один від одного;
- чим більше кількість аналізованих семплів, тим більш надійно може прийматись рішення;
- математичний та статистичний процес підтримує злиття даних;
- здатність отримувати інформацію від різнорідних пристроїв.

На рис. 1.7 представлений огляд процесу, що забезпечує проміжний сервер, який виконує просторовий та часовий аналіз. У просторовому аналізі використовується формула гаверсинуса для того, щоб визначити, які семпли

знаходяться в або поза діапазоном охоплення проміжного сервера. Часовий аналіз (Kruskal Wallis та алгоритм часового вікна) є фундаментальним для визначення, чи семпли надіслані всіма користувачами мають достатню кореляцію, іншими словами, чи насправді семпли вказують на сейсмічну небезпеку, про яку слід повідомити.

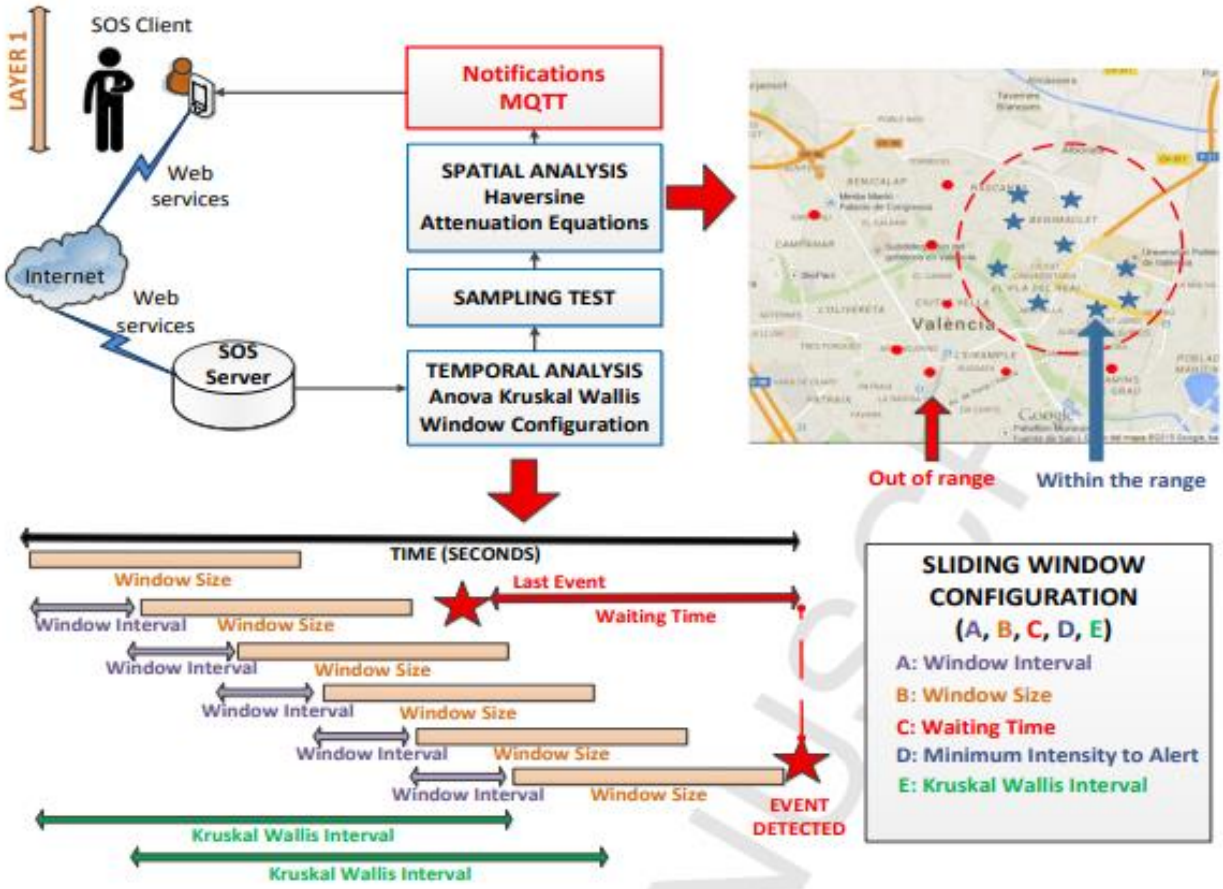

Рисунок 1.7 – Проміжний сервер: конфігурація розсувного вікна (A, B, C, D, E) (Крускал Валліс) та часовий та просторовий аналіз

Рівень 3 – центр управління поводиться як командно-контрольний пункт, надає інформацію про глобальні ризики надзвичайної ситуації відподвідним службам (пожежники, поліція, швидка допомога та іншим), допомагаючи їм приймати правильні рішення. Центр управління дозволяє розширити системи з управління та попередження перед подією на схему управління після події. Центр управління допомагає користувачам приймати рішення після події

надаючи «поради» щодо найближчих центрів допомоги, безпечніших та швидших маршрутів (так як користувач сам не знає про реальну ситуацію катастрофи). Все це відбувається через певний модуль тієї самої програми на смартфоні, який активується лише тоді, коли проміжний сервер перевірив та підтвердив наявність даного смартфону в межах землетрусу.

Отже, у третьому проаналізованому джерелі наведено приклад системи детектування землетрусів, сітка сенсорів якої – смартфони, якими користуються жителі сейсмоактивних регіонів. Великим плюсом є факт, що для такої системи, витрати на інфраструктуру будуть мінімальними. Недоліком є те, що потрібно дискримінувати корисні дані про сейсмічні піки від шумів через повсякденну нормальну користувацьку активність смартфоном. Також таке рішення здається не дуже придатним для регіонів з низькою густиною населення.

Проаналізувавши 3 аналогічні проекти, я виділив головні переваги та недоліки, які наведено у таблиці 1.1.

Таблиця 1.1 – Висновок огляду аналогів

Параметри	**Аналог 1**	**Аналог 2**	**Аналог 3**
Назва	Earthquake Early Warning System by IOT using Wireless Sensor Networks	A Smart IoT Device for Detecting and Responding to Earthquakes	Technologies of Internet of Things applied to an Earthquake Early Warning System
Країна	Індія	Південна Корея	Іспанія
Переваги	Добре описано процес тестування та валідації приладу сейсмографу.	Алгоритм детектування, що базується на нейромережі. Хороші приклади побудови IoT екосистеми на прикладі домашньої автоматики	Рішення виконано як додаток для смартфону, та використовує наявні ресурси смартфону для детектування сейсмоактивності.

Продовження табл. 1.1

Параметри	Аналог 1	Аналог 2	Аналог 3
Недоліки	Невдалий вибір протоколу зв'язку.	Рішення підходить тільки для домашньої автоматики – оскільки для передачі даних використовується Wi-Fi.	Низька густина покриття сенсорів у малонаселених районах. Високий показник шуму через коритувацьку активність.

З кожного проекту можна взяти щось корисне з точки зору реалізації приладів-сейсмографів, та алгоритму детектування сейсмічних піків.

1.3 Визначення вхідних даних для вирішення поставленої задачі

1.3.1 Поняття сейсмічних хвиль

Сейсмічні хвилі – це хвилі енергії, спричинені раптовим руйнуванням гірських порід у землі або вибухом. Вони є енергією, яка подорожує по землі і реєструється на сейсмографах.

Існує кілька різних видів сейсмічних хвиль, і всі вони рухаються по-різному. Два основних типи хвиль – це глибинні хвилі і поверхневі хвилі. Глибинні хвилі можуть подорожувати внутрішніми шарами землі, але поверхневі хвилі можуть рухатися лише вздовж поверхні планети, як брижі на воді. Землетруси випромінюють сейсмічну енергію як глибинними, так і поверхневими хвилями.

Подорожуючи товщею Землі, глибинні хвилі надходять швидше за поверхневі хвилі. Ці хвилі мають вищу частоту, ніж поверхневі.

Перший тип сейсмічних хвиль – це хвиля P або первинна хвиля. Це найшвидший вид сейсмічної хвилі і, отже, перший, що «прибуває» на сейсмограф. Хвиля P може рухатися через тверду породу та рідини, як вода або рідкі шари землі. Хвиля штовхає і тягне скелю, через яку рухається, так само, як звукові хвилі штовхають і тягнуть повітря. Вікна брязкають, бо звукові хвилі

штовхають і тягнуть за собою скло вікна, подібно до того, як хвилі P штовхають і тягнуть породу. Деякі тварини можуть відчувати P-хвилі землетрусу. Наприклад, собаки, як правило, починають істерично гавкати безпосередньо перед тим, як землетрус «вдарить» (або, точніше, до приходу поверхневих хвиль). Зазвичай люди можуть відчути лише удар і брязкання твердих тіл внаслідок цих хвиль.

Хвилі P також відомі як компресійні хвилі через їх напям поширення (штовхання та витягування). Піддаючись дії P-хвилі, частинки рухаються в тому самому напрямку, в якому рухається хвиля, тобто напрямку, в якому рухається енергія (рис. 1.8).

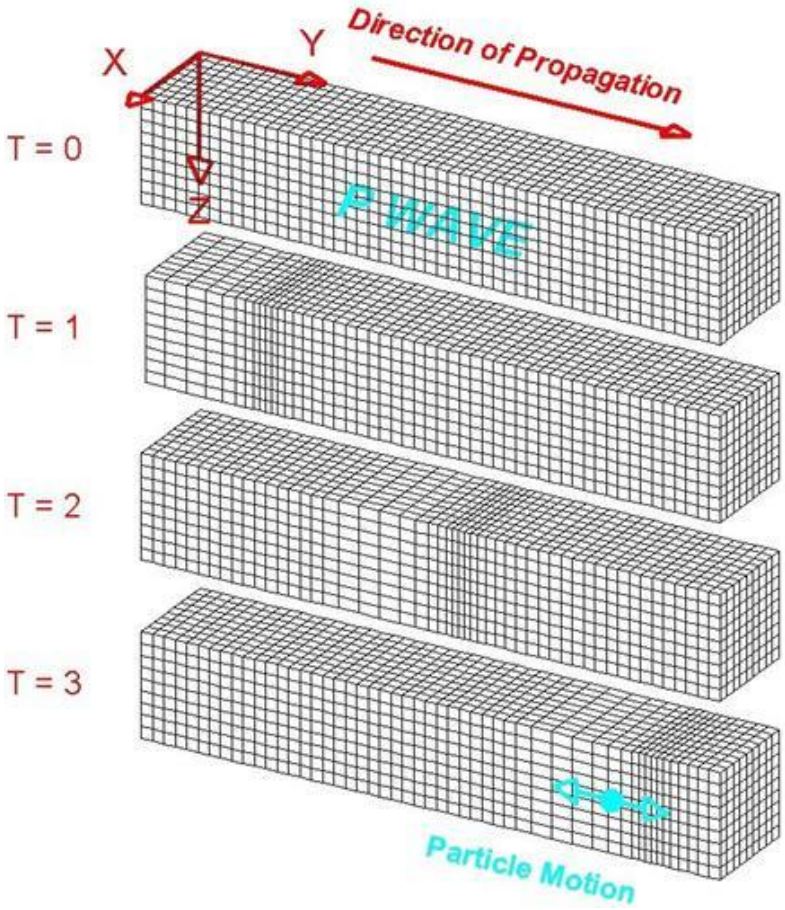

Рисунок 1.8 – Розповсюдження P-хвилі [12]

Другий тип глибинної хвилі – це хвиля S або вторинна хвиля, яка є другою хвилею, яку ви відчуваєте під час землетрусу. Хвиля S повільніша, ніж хвиля P, і може рухатися лише через тверду породу, а не через рідке середовище. Саме ця властивість S-хвиль привела сейсмологів до висновку, що зовнішнє ядро Землі - це рідина. S-хвилі рухають частинки гірських порід вгору і вниз, або в сторону – перпендикулярно напрямку, в якому рухається хвиля (напрямку поширення хвилі) (рис. 1.9).

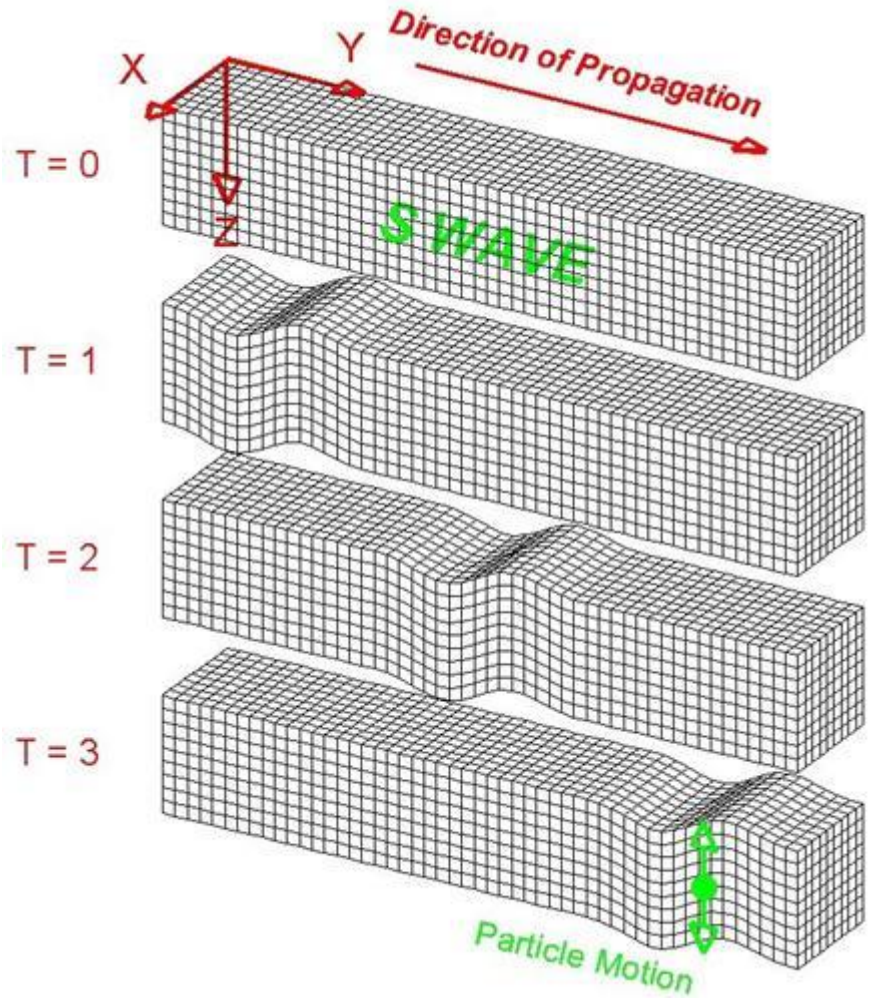

Рисунок 1.9 – Розповсюдження S-хвилі [12]

Сенсори мають реєструвати P і S хвилі.

1.3.2 Синхронізація часу в IoT

Існує проблема синхронізації часу у сейсмографів та інших підсистем. Сенсори мають реєструвати коливання з високою точністю у часі. Останні досягнення в галузі вбудованого інтелекту, зв’язку та технологій взаємодії дозволило інтегрувати об'єкти з нашого повсякденного життя в комунікаційні мережі для взаємодії один з одним через Інтернет. Широкі можливості взаємозв’язку, підтримувані IoT, створюють проблему взаємодії між різними об'єктами з гетерогенними можливостями [13]. У типовій архітектурі IoT, беруть участь три різні мережі, як показано на рис. 1.10. Вузол глобальної мережі (WAN), наприклад, користувацький девайс стільникової мережі підключений до вузла локальної мережі (LAN) через контролер мережевого інтерфейсу (NIC), і цей вузол підключений до персональної мережі (PAN) технологією бездротового зв'язку з низьким енергоспоживанням і коротким діапазоном. Деякі з цих об'єктів можуть мати велику кількість обчислювальних та комунікаційних ресурси, деякі можуть бути енергообмеженими вузлами бездротових датчиків, а деякі можуть бути пасивними простими пристроями, такими як теги RFID. Хоча ці компоненти можуть бути взаємопов’язані за допомогою сучасних Інтернет-технологій, не всі об’єкти можуть містити достатньо ресурсів, необхідні для реалізації цих рішень. Один із підходів полягає у використанні девайсу, що з’єднує декілька шарів архітектури на протокольних рівнях між пристроями та деякий девайс для підключення до Інтернету, наприклад, шлюз [14]. Однак такий підхід вводить додаткові витрати на обробку та конвертацію, що зменшує ефективність додатків, що вимагають жорстких вимог взаємодія з фізичним світом.

На рис. 1.10 ліворуч зображено вузол глобальної мережі (WAN) що підключений до Інтернету через контролер мережевого інтерфейсу (NIC) і підключений до локальної мережі (LAN) (у центрі рис. 1.10) за допомогою іншої мережевої карти. Один з вузлів локальної мережі підключений до персональної мережі (PAN) праворуч, що реалізує сенсорну мережу. Кожен процесор в

архітектурі має своє уявлення про час, внаслідок різних реалізацій системного годинника [16].

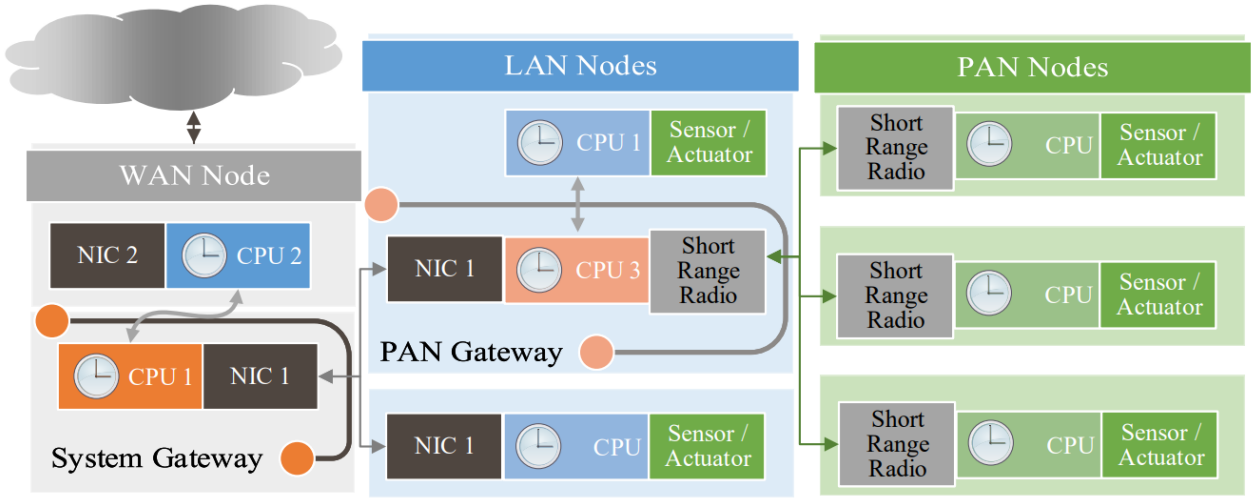

Рисунок 1.10 – Типова архітектура Інтернету речей з використанням бездротової сенсорної мережі для взаємодії з фізичним світом. [15]

На рис. 1.10 ліворуч зображено вузол глобальної мережі (WAN) що підключений до Інтернету через контролер мережевого інтерфейсу (NIC) і підключений до локальної мережі (LAN) (у центрі рис. 1.10) за допомогою іншої мережевої карти. Один з вузлів локальної мережі підключений до персональної мережі (PAN) праворуч, що реалізує сенсорну мережу. Кожен процесор в архітектурі має своє уявлення про час, внаслідок різних реалізацій системного годинника [16].

Бездротові сенсорні мережі (WSN) [17] є однією з технологій IoT. Вони передають отримані цифрові дані з фізичного світу в Інтернет або навпаки отримують дані з Інтернету, що описують дії щодо внесення деяких змін в оточення без втручання людини [18]. Це досягається передаванням інформаційного змісту виконуючи різні конвертації та змінюючи протоколи між задіяними об'єктами зв'язку та обробки. У зв'язку з цим програми, що вимагають хронологічного впорядкування інформації або синхронного виконання для синтезу даних або планування передачі даних з розподілом часу [19] з низьким енергоспоживанням вимагають узгоджене поняття часу, який має бути спільним

для всіх об'єктів, що беруть участь як в обробці, так і в спілкуванні. Поняття часу у кожній системі відрізняється від загального поняття часу через кілька факторів [20]. Вплив цих помилок нівелюється за рахунок:

- передача звіту про час контрольного годинника в системі за допомогою протоколу обміну повідомленнями;
- пом'якшення недетермінованості доставки повідомлень та вимірювання часу за допомогою алгоритму часової дисципліни (CDA);
- періодичного налаштування годинників.

Для типової IoT архітектури існує три різні мережі, які мають різні поняття часу, як показано на рис. 1.10 . Для глобальних мереж та локальних мереж проблема синхронізації часу добре вирішується такими рішення, як Network Time Protocol (NTP) [21] (або його спрощена версія – Simple Network Time Protocol [22]) та протокол точності часу (PTP) [23]. Наприклад, вузол WAN (а також вузли локальної мережі) можуть синхронізуватися з часом в Інтернеті, наприклад, за допомогою NTP. Однак такі синхронізації не завжди можуть бути виконані підсистемами з обмеженими енергетичними та обчислювальними ресурсами, і такі підсистеми вимагають більш простих методів. Існуюче рішення для синхронізації часу у WSN розроблені для обмежених пристроїв (наприклад, мережі PAN на рис. 1.10), але такі реалізації підходять для конкретного сценарію застосування. Тому для кожної архітектури IoT потрібно вибрати відповідне рішення для синхронізації часу з урахуванням сценарію застосування та можливостей мережевих об'єктів для досягнення необхідного рівня продуктивності.

1.4 Висновки після першого розділу

У розділі розглянуто концепції, на яких будуються системи раннього сповіщення, порівняно аналогічні системи. Проаналізовані вдалі конструктивні рішення, які в подальшому будуть використовуватись у розробці IoT системи

моніторингу. Наведено основні теоретичні відомості підходів, що будуть використовуватись у проекті. Також визначено вхідні дані для вирішення задачі магістерської роботи.

Розділ 2. Розробка проекту IoT-рішення та методів обробки даних

2.1 Розробка проекту IoT-рішення

З технічного завдання робимо висновок, що потрібно розробити IOT систему для моніторингу сейсмологічної ситуації Закарпатського регіону. Система буде використовуватись для раннього попередження жителів та туристів регоіну про землетруси. Аналізуючи вимогу, було прийнято рішення використовувати SMS-сповіщення жителів регіону, та push-повідомлення. Останній метод дозволяє не опиратись на дані геолокації, використовуючи мобільні оператори, оскільки, з будь-яких причин такі дані можуть не надаватись оператором. Використовуючи додаток для мобільних платформ, можна отримувати доступ до геолокації смартфону, а користувачі можуть налаштовувати бажаний поріг інформування про землетруси, якщо незначні коливання їх не цікавлять.

В завдання входить розробка датчику-сейсмографу для детектування землетрусів, та так званого, центру обробки і аналізу, що прийматиме дані від деякої кількості сейсмографів. Сейсмографи мають містити у собі сенсори достатньої чутливості щоб реєструвати механічні коливання по двом або трьом осям. Сенсори мають реєструвати P і S хвилі.

Сейсмограф має працювати автономно декілька місяців. Так як у технічному завданні не вказано ємність акумулятору, будемо відштовхуватись від струму споживання, намагатимемось його зменшити, щоб покращити час роботи на одному заряді. Для цього будемо використовувати режим низької споживаної потужності у мікропроцесорі. Вважатимемо одну з вимог при виборі мікропроцесора наявність такого режиму. Розробку програмного забезпечення для мікропроцесора, що буде використовуватись у датчику-сейсмографі потрібно виконувати також з міркувань низького енергоспоживання.

Розміщуватись сейсмограф має на поверхні землі, або на невеликій глибині під поверхнею, тому кліматичне виконання сейсмографу – УХЛ1.1 за ГОСТ 15150-69. Під таким кліматичним виконанням вважається працездатність

пристрою у помірному і холодному макрокліматичних районів, на відкритому повітрі з впливом будь-яких атмосферних факторів (дощ, злива, сніг, пил при сильному вітрі). Хоч тема корпусування не буде розкрита у цій роботі, підбір деталей потрібно здіснити зважаючи на умови експлуатації.

2.2 Методи та засоби обробки даних

2.2.1 Алгоритм короткострокового та довгострокового усереднення

Алгоритм короткострокового та довгострокового усереднення (short-time-average/long-time-average – STA/LTA) зазвичай використовується в системах моніторингу сейсмічної активності із слабкими амплітудами поштовхів, які намагаються записати якомога більше сейсмічних подій. Це системи, де алгоритм STA/LTA є найбільш корисним. Це майже стандартний алгоритм в портативних сейсмічних реєстраторах, а також у багатьох програмних пакетах для обробки в реальному часі мереж сейсмічних сенсорів, у регіонах зі слабкою активністю. Однак він також може бути корисним у багатьох випадках сильного сейсмічного руху, за винятком випадків, коли реєструвати потрібно виключно найсильніші землетруси. Тригер за допомогою (STA/LTA) значно покращує реєстрацію слабких землетрусів у порівнянні з алгоритмами де відбувається спрацьовування по порогових значеннях амплітуди. Одночасно це зменшує кількість помилкових семплів, спричинених природним та техногенним сейсмічним шумом. Певною мірою алгоритм також допускає дискримінацію різних видів землетрусів. Налаштування параметрів тригера STA/LTA завжди є компромісом серед кількох сейсмологічних та інструментальних міркувань. Метою пошуку оптимальних параметрів є максимально висока чутливість сейсмічної станції для даного типу сейсмічного сигналу.

Тригер STA/LTA є найбільш корисним на сейсмічно тихих ділянках, де природний сейсмічний шум є домінуючим типом сейсмічного шуму. Він також ефективний у разі мінливого «безперервного» техногенного сейсмічного шуму. Така мінливість, наприклад, відбувається внаслідок зміни денної/нічної людської діяльності поблизу або в міських районах. Алгоритм STA/LTA є менш

ефективним за наявності нерегулярних високоамплітудних техногенних сейсмічних шумів, які часто бувають поодинокими «стрибками», або серією із «стрибків».

Алгоритм STA/LTA постійно відстежує постійні зміни амплітуди сейсмічного шуму на ділянці де встановлено сейсмограф та автоматично регулює чутливість акселерометру до фактичного рівня сейсмічного шуму. В результаті досягається значно вища чутливість системи під час сейсмічно тихих періодів, а надмірна кількість помилково спрацьованих записів нівелюється або, принаймні, пом'якшується, під час сейсмічно активних періодів. Обчислення неодноразово в режимі реального часу. Цей процес, як правило, відбувається незалежно в усіх сейсмографах в мережі.

Алгоритм STA/LTA обробляє фільтровані сейсмічні семпли в двох рухомих часових вікнах – вікні короткого часу (STA) та вікні тривалого часу (LTA). STA вимірює "миттєву" амплітуду сейсмічного сигналу та спостерігає за землетрусами. LTA дбає про поточну середню амплітуду сейсмічного шуму.

Спочатку обчислюється абсолютна амплітуда кожної вибірки даних вхідного сигналу. Далі обчислюється середнє значення абсолютних амплітуд в обох вікнах. На наступному етапі розраховується співвідношення обох значень – співвідношення STA/LTA. Цей коефіцієнт постійно порівнюється з вибраним користувачем пороговим значенням – пороговим рівнем спрацьовування STA/LTA. Якщо коефіцієнт перевищує цей поріг, спрацьовує тригер каналу. Тригер каналу не обов'язково означає, що реєстратор даних або мережа фактично починає реєструвати сейсмічні сигнали. Усі сейсмічні мережі та більшість сейсмічних реєстраторів мають вбудований механізм "тригерного голосування", який визначає, скільки і які канали повинні бути у спрацьованому стані до того, як прилад або мережа фактично почнуть записувати дані. Щоб спростити пояснення, ми будемо спостерігати лише один сигнальний канал. Будемо вважати, що тригер каналу еквівалентний тригеру мережі або одного сейсмографу.

Після того, як сейсмічний сигнал поступово припиняється, тригер каналу вимикається. Це трапляється, коли поточне співвідношення STA/LTA опускається нижче іншого вибраного користувачем параметра – порогового рівня відключення STA/LTA. Очевидно, що пороговий рівень відключення STA/LTA повинен бути нижчим (або, дуже рідко, рівним), ніж пороговий рівень спрацьовування STA/LTA.

На додаток до даних, отриманих під час "активації тригера", сейсмічні мережі та сейсмічні реєстратори додають певний обсяг сейсмічних даних до файлу подій перед активацією – дані до події (ДДП). Після завершення активного стану тригера також додаються дані після події (ДПП).

Для кращого розуміння на рис. 2.1 наведено типову локальну подію та змінні тригера (спрощені) під час спрацьовування STA/LTA. Графік a) показує вхідний безперервний сейсмічний сигнал (відфільтрований); графік b) показує усереднений абсолютний сигнал у часових вікнах STA та LTA відповідно, коли вони рухаються в часі; і графік c) показує співвідношення обох. Крім того, відображаються активний стан тригера (суцільний прямокутник), час після події (ДПП) та час до події (ДДП) (прямокутники з пунктирною лінією). У цьому прикладі параметру порогового рівня тригера було встановлено значення 10, а пороговому рівню детригера значення 2 (дві горизонтальні пунктирні лінії). Можна помітити, що тригер став активним, коли значення коефіцієнта STA/LTA перевищило значення 10. Також він був деактивований, коли значення коефіцієнта STA/LTA опустилося нижче значення 2. На графіку d) відображається фактично записані дані. Вони включають всі важливі етапи подій та частину сейсмічного шуму на початку.

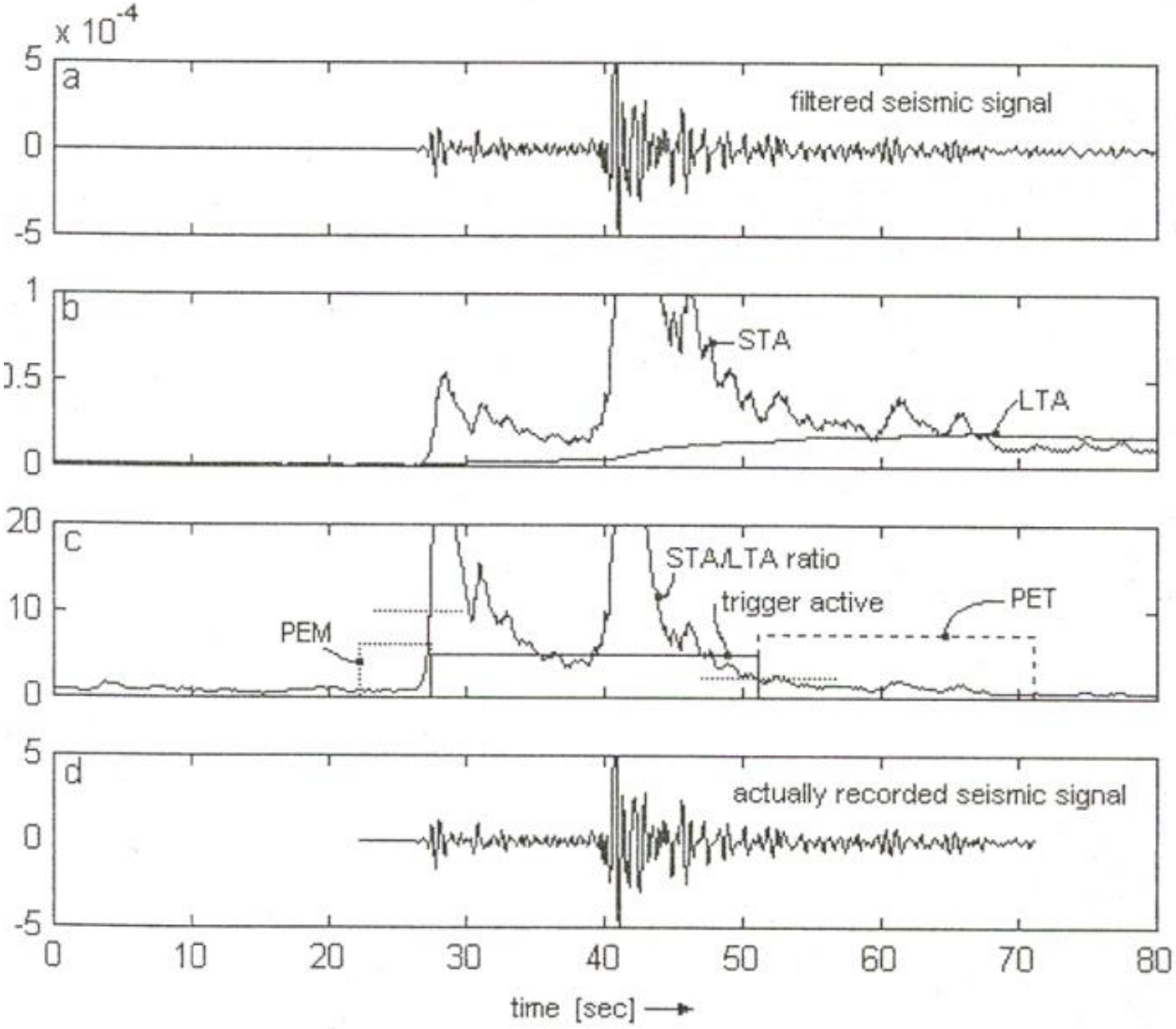

Рисунок 2.1 – Функція та змінні обчислень тригера STA / LTA [24]

Насправді тригери STA/LTA зазвичай дещо складніші, однак деталі не є важливими для розуміння та правильної параметризації тригера.

2.2.1.1. Налаштування параметрів часових вікон та тригеру

Для встановлення основних параметрів алгоритму спрацьовування STA/LTA потрібно вибрати наступне:

- тривалість вікна STA;
- тривалість вікна LTA;
- пороговий рівень спрацювання тригера STA/LTA;
- пороговий рівень відключення STA/LTA.

Однак оптимальне спрацьовування сейсмічного реєстратора або сейсмічної мережі не залежить лише від цих параметрів. Зазвичай є чотири додаткові пов'язані параметри. Якщо вони добре налаштовані, разом з параметрами тригера, гарантують оптимальний запис даних. Це параметри:

- тригерні фільтри;
- дані за час до події (ДДП);
- дані за час після події (ДПП);
- схема голосування за тригером.

Хоча ці додаткові параметри безпосередньо не пов'язані з алгоритмом STA/LTA, ці додаткові параметри також обговорюються нижче, щоб надати повну інформацію.

Параметри тригера STA/LTA та відповідні параметри залежать від цілі програми, від стану сейсмічного шуму на ділянці, від властивостей сейсмічних сигналів у даному місці та від типу використовуваного датчика. Усі ці проблеми в широкій мірі різняться між застосуваннями та між сейсмічними зонами. Очевидно, що загального, єдиного правила щодо їх встановлення не існує. Кожне застосування та кожна сейсмічна зона вимагає певного вивчення, оскільки лише практичний досвід дозволяє визначити дійсно оптимальні налаштування алгоритму.

Зауважимо, що сейсмічні реєстратори та програмні пакети для мережі сенсорів постачаються із набором тригерів за замовчуванням (заводські налаштування) та відповідних значень параметрів. Вони рідко є оптимальними, тому їх слід коригувати, щоб вони були ефективними в конкретному застосуванні. Для досягнення найкращих результатів зміна цих параметрів та поступовий пошук найкращих налаштувань – це процес, який вимагає певних зусиль та часу.

2.2.1.2. Підбір тривалості часового вікна короткострокового усереднення (STA)

Часове вікно короткострокового усереднення містить "миттєве" значення виміряного сейсмічного сигналу або його огинаючої. Як правило, тривалість STA повинна бути більше кількох періодів типово очікуваного сейсмічного сигналу. Якщо STA занадто короткий, усереднення сейсмічного сигналу не працюватиме належним чином. STA більше не є мірою середнього сигналу (огинаючої сигналу), так як на нього впливають окремі періоди сейсмічного сигналу. З іншого боку, тривалість STA має бути коротшою за найкоротші сейсмічні події, які ми очікуємо зафіксувати.

Певною мірою STA функціонує як сигнальний фільтр. Чим коротша вибрана тривалість, тим вища чутливість тригера до короткочасних локальних землетрусів у порівнянні з тривалими та менш частотними, віддаленими землетрусами. Чим довша вибрана тривалість STA, тим менш вона чутлива до короткочасних локальних землетрусів. Тому, змінюючи тривалість STA, можна певною мірою визначити пріоритет фіксування віддалених або наближених подій.

Тривалість вікна STA також важлива щодо попередження помилкових спрацювань. Зменшуючи тривалість вікна STA, спрацьовування стає більш чутливим до техногенних сейсмічних шумів, і навпаки. Хоча такий шум зазвичай має інструментальний характер, він також може мати сейсмічний характер. У зонах, сильно забруднених стрибкоподібними шумами, зазвичай потрібно обирати тривалість STA значно більшою, ніж ці стрибки, якщо помилкових тригерів занадто багато. На жаль, це також зменшить чутливість запису до дуже локальних короткочасних подій. На рис. 2.2 зображено вплив тривалості STA на місцеві події та стрибкоподібного шуму.

На графіку а) показаний сигнал з штучним стрибком ліворуч і з коротким, дуже локальним землетрусом праворуч.

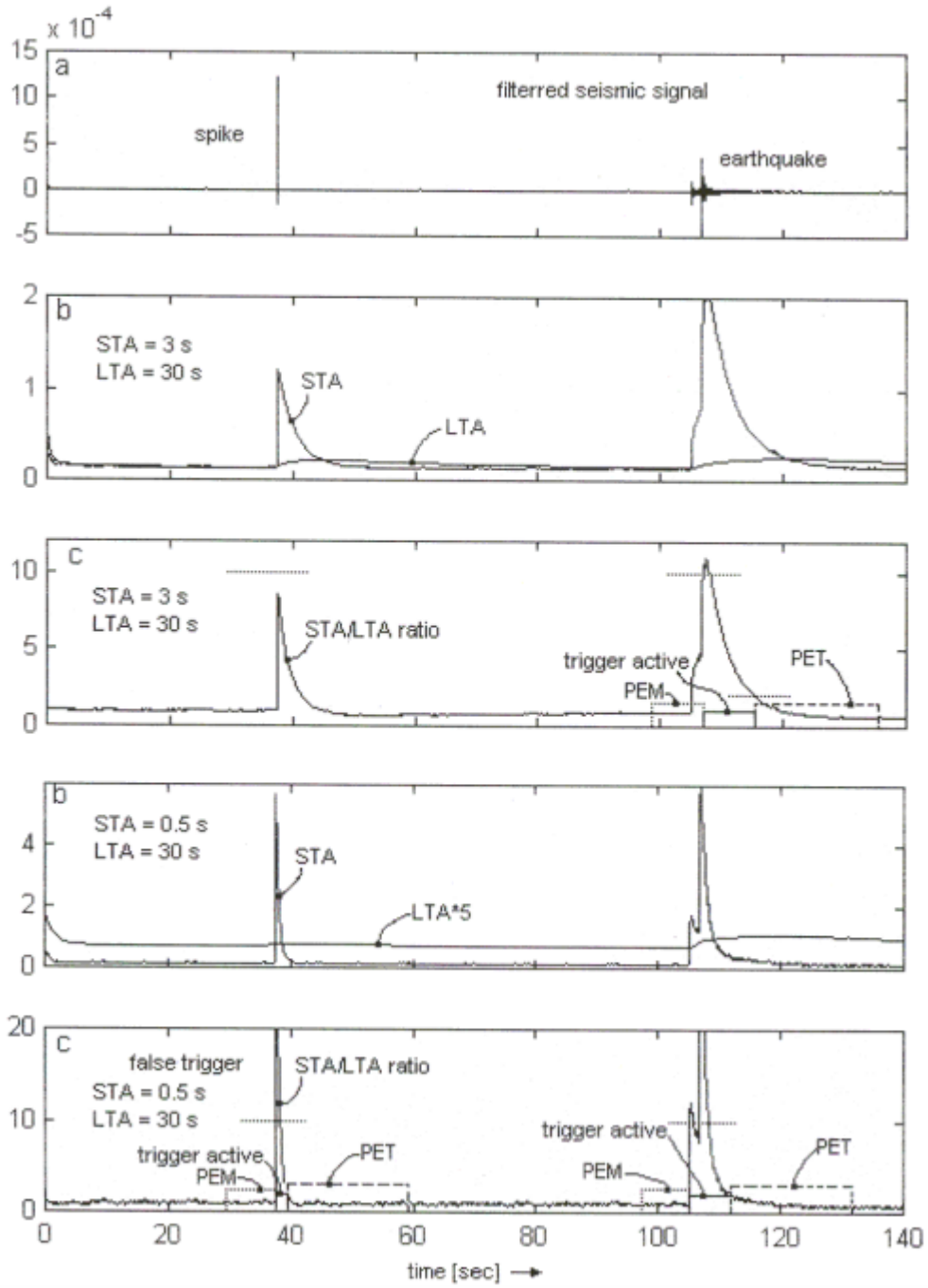

Рисунок 2.2 – Вплив тривалості STA на чутливість тригера до короткочасних місцевих подій та «стрибкоподібного» шуму в сейсмічних сигналах [24]

Графіки б) та в) показують співвідношення STA, LTA, STA/LTA та активні стани тригеру разом з ДДП та ДПП. Поріг спрацьовування STA/LTA був встановлений на значення 10, а поріг детригера – на значення 2. Можна помітити, що при використанні відносно тривалого STA у 3 секунди землетрус змусив

систему спрацювати. Однак набагато більший за амплітудою (але коротший) інструментальний стрибок не спричинив спрацювання. Співвідношення STA/LTA не перевищувало порогове значення, і не було помилково спрацьованого запису через стрибок. Нижні два графіки показують ті самі змінні, але для коротшого STA на 0,5 сек. Стрибок явно спровокував роботу системи і спричинив помилковий запис. Звичайно, землетрус також спричинив роботу системи.

Для більш маштабних сейсмічних подій типове значення тривалості STA становить від 1 до 2 секунд. Для місцевих землетрусів на практиці зазвичай використовуються коротші значення близько 0,5 - 0,3 с.

2.2.1.3. Підбір тривалості часового вікна довгострокового усереднення (LTA)

Часове вікно LTA вимірює сейсмічний шум середньої амплітуди. Це вікно повинно тривати довше, ніж кілька періодів типово нерегулярних коливань сейсмічного шуму. Змінюючи тривалість вікна LTA, можна зробити запис більш-менш чутливим до регіональних подій в діапазоні хвиль P від 200 до 1500 км до епіцентру. Такими подіями, як правило, спочатку є низькоамплітудні хвилі P. Коротша тривалість вікна LTA дозволяє результуючій величині LTA дещо адаптуватися до повільно зростаючої амплітуди сейсмічних хвиль, що виникають. Таким чином, співвідношення STA/LTA залишається низьким, незважаючи на збільшення складової STA. Це ефективно зменшує чутливість тригера до таких подій. У протилежному випадку, використовуючи тривале часове вікно LTA, чутливість тригера до виникаючих землетрусів підвищується, оскільки на значення LTA не так швидко впливає виникаючий сейсмічний сигнал, що дозволяє хвилям P запускати запис.

На рис. 2.3 пояснюється описана ситуація. На графіку а) показана подія з виникаючими P-хвилями. Графіки б) і в) показують часовий хід параметрів тригера для відносно тривалої LTA 60 секунд. LTA не змінюється швидко, дозволяючи співвідношенню STA/LTA перевищувати поріг спрацьовування

(коротка горизонтальна пунктирна лінія). Графіки d) і e) показують ту ж ситуацію з коротшим LTA на 30 секунд. Значення LTA зростає набагато швидше під час початкової фази події, таким чином зменшуючи значення співвідношення STA/LTA, яке не перевищує порогового значення STA/LTA. Спрацьовування не відбувається, і подія пропущена.

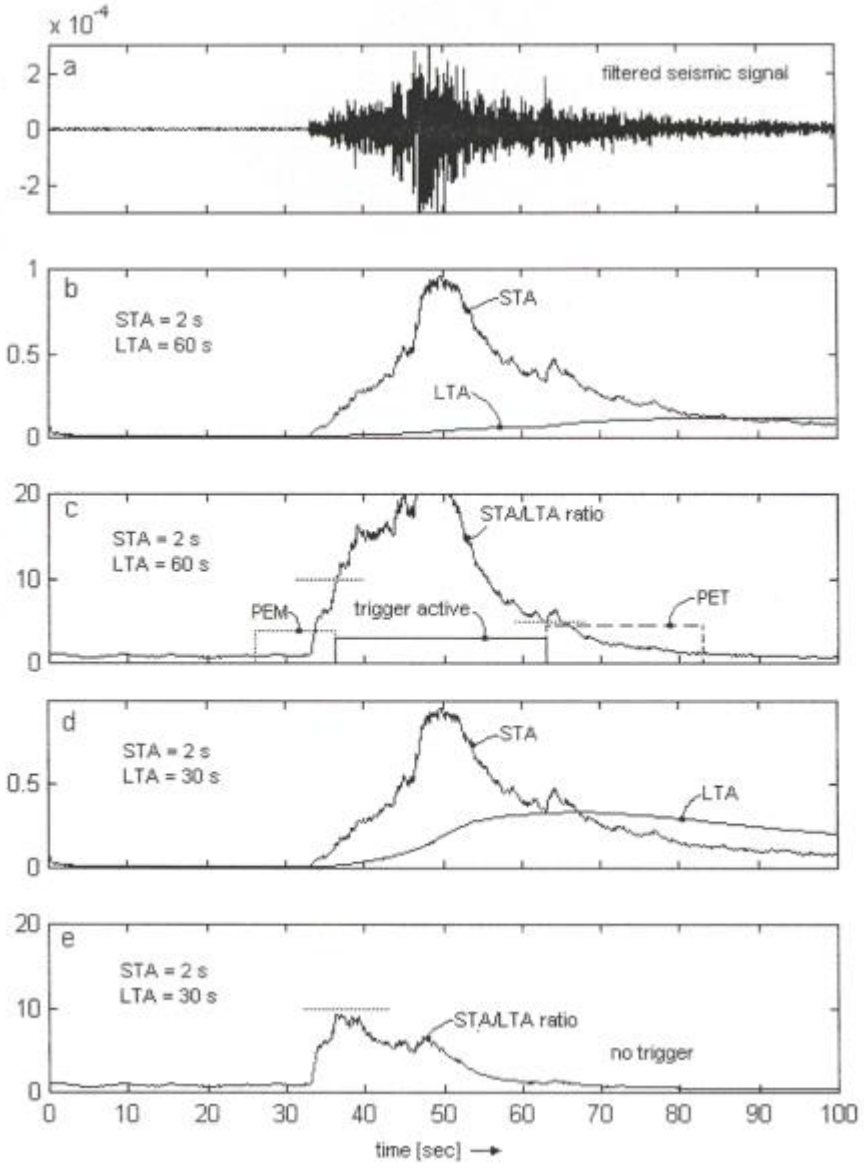

Рисунок 2.3 – Вплив тривалості STA на чутливість тригера до короткочасних місцевих подій та «стрибкоподібного» шуму в сейсмічних сигналах [24]

Подібним чином, ефективне спрацьовування запису подій із слабкими P-хвилями порівняно з S-хвилями вимагає більш тривалого вікна LTA з двох

причин. По-перше, якщо P-хвилі не спрацьовують, вони "забруднюють" справжню інформацію про сейсмічний шум до події, виміряної за час LTA, оскільки їх амплітуда перевищує амплітуду сейсмічного шуму до події. Це призводить до зменшення чутливості тригера в той момент, коли надходять S-хвилі. Це "забруднення" зменшується, якщо обрано більшу тривалість LTA. По-друге, довший LTA робить спусковий механізм більш чутливим і до P-хвиль, якщо вони не мають суто ударного типу.

На рис. 2.3 представлений такий випадок. Графік a) показує типову подію із значно більшими пізнішими фазовими S хвилями, ніж швидкі P-хвилі. Графіки b) і c) показують параметри тригера для тривалої LTA 100 секунд. Послідовність P-хвиль, а також послідовність S-хвиль діють на реєстратор. Відповідні ДДП та ДПП фіксуються повністю в одному файлі з усіма сейсмічними фазами та частинами сейсмічного шуму перед ними. Графіки d) і e) показують ту ж ситуацію, але для коротшого LTA в 45 секунд. Можна помітити, що хвилі P взагалі не запустили запис, тоді як хвилі S ледве спровокували тригер. Співвідношення STA/LTA навряд перевищує пороговий рівень STA/LTA. Як результат, записаний файл даних є занадто коротким. У цьому записі відсутні P-хвилі та інформація про сейсмічні шуми перед ними. Трохи менша подія взагалі не спричинить запис.

З іншого боку, використання короткого вікна LTA забезпечить достатню чутливість реєстратора до поступових змін «безперервного» техногенного сейсмічного шуму. Такий "перехід" техногенного сейсмічного шуму від низької до високої інтенсивності типовий для нічного переходу людської діяльності в міських районах. Іноді, використовуючи коротке вікно LTA, можна пом'якшити помилкові тригери системи через транспортний трафік. Прикладами таких випадків може бути важкий транспортний засіб, що під'їжджає та проходить поруч із сейсмічною станцією на місцевій дорозі. Короткий LTA може "пристосуватися" досить швидко для таких виникаючих активностей запобігти помилковим тригерам.

На рис. 2.4 наведено приклад реакції LTA на підвищений сейсмічний шум. Графік а) показує сейсмічний шум, інтенсивність якого поступово збільшувалася в середині запису.

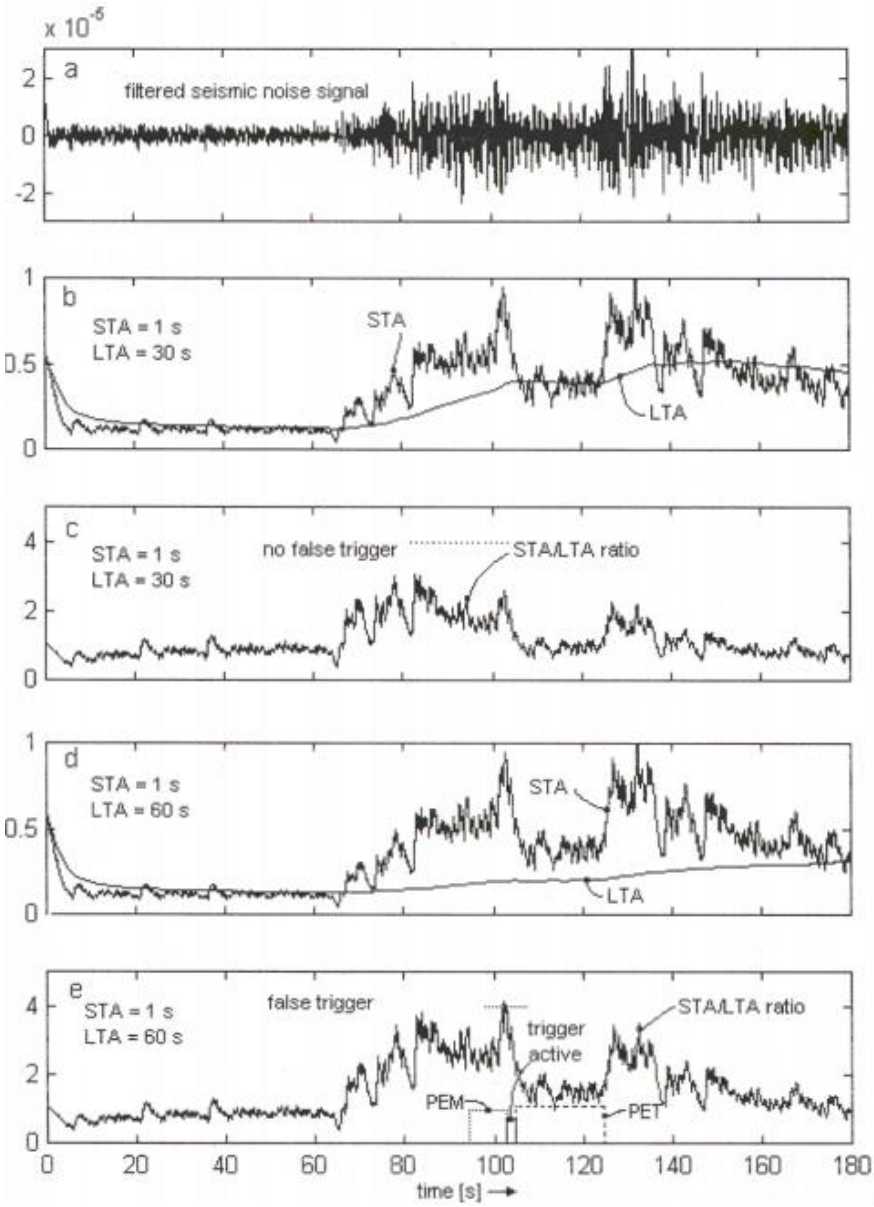

Рисунок 2.4 – Вплив тривалості LTA на помилкове спрацьовування при зміні умов сейсмічного шуму [24]

Звернемо увагу, що зміна його амплітуди не є раптовою, а триває приблизно 20-30 секунд. Графіки б) і в) показують ситуацію при короткому LTA 30 секунд. Можна помітити, що значення LTA більш-менш відстежує збільшену амплітуду шуму. Співвідношення STA/LTA залишається набагато нижчим за

поріг спрацьовування, і помилкового спрацьовування не відбувається, незважаючи на значно підвищений сейсмічний шум на ділянці. Графіки d) та e) показують ситуацію з довшим LTA на 60 секунд. У цьому випадку LTA не змінюється надто швидко, що дозволяє підвищити співвідношення STA/LTA під час збільшення шуму. В результаті виникає помилковий тригер і генерується помилковий запис.

Природний сейсмічний шум (морський шум) може змінювати свою амплітуду в два рази, перевищуючи величину спрацювання. Однак ці зміни повільні. Суттєві зміни можуть відбутися лише протягом декількох годин, а в гіршому – декількох десятків хвилин. Тому навіть найдовша тривалість LTA є недостатньою, щоб LTA міг повністю відповідати варіаціям амплітуди морського шуму. Тривалість LTA 60 секунд є загальним початковим значенням. Потрібна менша тривалість LTA, щоб виключити виникнення регіональних подій, якщо це бажано, або якщо для ділянки характерним є швидко мінливий техногенний шум. Більш тривалий LTA може бути використаний для віддалених подій з дуже великим часом між хвилями S та P.

2.2.1.4. Підбір порогового рівня спрацьовування STA/LTA (тригера)

Пороговий рівень спрацьовування STA/LTA найбільшою мірою визначає, які події будуть записані, а які ні. Чим більше значення встановлюється, тим більше землетрусів вірогідно не будуть зафіксовані, але буде менше помилкових спрацьовувань. Чим нижчим вибрано пороговий рівень спрацьовування STA/LTA, тим більш чутливою буде сейсмічна станція і тим більше подій буде записано. Однак частіші помилкові тригери також займуть пам'ять даних та обтяжать аналітику. Оптимальний пороговий рівень спрацьовування STA/LTA залежить від умов сейсмічного шуму на ділянці та від толерантності до помилково спрацьованих записів. На встановлення оптимального порогового рівня спрацьовування STA/LTA впливає не тільки амплітуда, але й тип сейсмічного шуму. Статистично стаціонарний сейсмічний шум (з меншими

нерегулярними коливаннями) дозволяє знизити пороговий рівень тригера STA/LTA.

Вибір одночасно дуже високого порогового рівня тригера STA/LTA та високого коефіцієнта підсилення каналу є недоцільним. Багато сейсмографів на ринку мають цю настройку ввімкнутою за замовчуванням без попереджувальних повідомлень. Ця ситуація особливо небезпечна в надзвичайно шумних умовах, де через занадто велику кількість помилкових спрацьовувань інструменти, як правило, встановлюються для запису лише найсильніших подій.

Припустимо, встановлено пороговий рівень спрацьовування STA/LTA на 20. Припустимо також, що коефіцієнт підсилення каналу обрано таким чином, що він має близько 150 мВ середнього сигналу сейсмічного шуму на вході реєстратора і повний діапазон напруги на вході каналу становить ± 2,5 В. Очевидно, що для цього налаштування потрібна буде амплітуда сигналу 0,15 В × 20 = 3 В для запуску каналу. Оскільки його максимальна вхідна амплітуда обмежена 2,5 В, вона ніколи не може спрацьовувати, незалежно від того, наскільки сильним є землетрус.

З деякими продуктами на ринку ця потенційна небезпека помилкового налаштування вирішується наступним чином: кожен раз, коли використовується алгоритм STA/LTA, додатковий пороговий алгоритм тригера залишається активним як фоновий процес. Через це канал спрацьовує, коли його вхідна амплітуда перевищує 50% від діапазону вхідної напруги каналу, не зважаючи на налаштування тригера STA/LTA. Таким чином, записуються найсильніші, а отже, найважливіші події, незалежно від того, наскільки необережно встановлюються параметри алгоритму тригера STA/LTA.

Початкове налаштування для порогового рівня спрацьовування STA/LTA 4 є загальним для середньої тихої сейсмічної ділянки. Значно нижчі значення можна застосовувати лише на станціях з відсутнім або дуже малим значенням штучного сейсмічного шуму. Більш високі значення близько 8 і вище необхідні на менш сприятливих ділянках із значним техногенним сейсмічним шумом.

2.2.1.5. Підбір порогового рівня детригера STA/LTA

Пороговий рівень детригера STA/LTA визначає рівень сигналу, при якому відбувається припинення запису даних. Пороговий рівень відхилення STA/LTA визначає, наскільки добре будуть зафіксовані дані про хвилі землетрусів. Щоб включити якомога більше хвиль, потрібно обирати низьке значення. Однак занадто низький рівень порогового значення STA/LTA часом є небезпечним. Це може спричинити дуже довгі або навіть нескінченні записи, наприклад, якщо раптове збільшення сейсмічного шуму не дозволяє співвідношенню STA/LTA опускатися нижче порогового рівня детригера STA/LTA.

Взагалі, чим шумнішою є сейсмічна ділянка, тим вище значення порогового рівня детригера STA/LTA слід використовувати для запобігання занадто довгих або безперервних записів. Ця небезпека висока лише на ділянках, сильно забруднених техногенним сейсмічним шумом.

Типовим початковим значенням порогового рівня відхилення STA/LTA є 2-3 для сейсмічно тихих ділянок. Для більш шумних місць необхідно встановлювати вищі значення.

2.2.1.6. Практичні рекомендації для знаходження оптимальних порогових рівнів

Для успішного коригування оптимального спрацьовування та пов'язаних з ним параметрів необхідний систематичний підхід. По-перше, цілі сейсмічної установки повинні бути ретельно продумані та враховані актуальні дані про сейсмічний шум (якщо такий є) на ділянках. На основі цієї інформації встановлюються початкові параметри. Інформація про них повинна зберігатися для документування. Рекомендовано починати із досить низьких налаштувань порогового рівня запуску, ніж із занадто високих. В іншому випадку можна витратити занадто багато часу на отримання достатньої кількості записів для змістовного аналізу, необхідного для подальших кроків коригування. Потім прилад або мережа залишаються працювати протягом певного періоду часу. Необхідна тривалість експлуатації без зміни параметрів запису сильно залежить

від сейсмічної активності в регіоні. Потрібно записати щонайменше кілька землетрусів та (або) помилково спрацьованих записів, перш ніж здійснити першу корекцію параметрів. Судження, засновані на одному або кількох записах, рідко призводять до поліпшень. Така робота просто не приносить жодного значущого коригування. Потім усі записи, включаючи помилково спрацьовані, повинні бути перевірені. Перевіряється повнота записів подій (сейсмічний шум, надходження P хвилі) та аналізуються причини помилкових тригерів. Співвідношення записів подій до помилкових записів розраховується та порівнюється з цільовим рівнем. Якщо кількість помилкових тригерів не досягає прийнятого рівня, потрібно збільшити чутливість тригера, знизивши поріг (пороги) тригера STA/LTA. В основному, ми отримуватимемо більше сейсмічної інформації майже за однакову ціну та зусилля. Якщо кількість помилкових тригерів занадто висока, знайдіть причини і спробуйте пом'якшити їх, змінивши параметри STA / LTA та (або) схеми голосування. Тільки якщо це не допомагає, слід зменшувати чутливість тригера. Після завершення аналізу параметри змінюються відповідно до його висновків та нових налаштувань, що архівуються для цілей документації. Знову прилад або мережа залишаються активними протягом певного періоду, аналізуються нові записи та за потреби вносяться інші зміни. Повторюючи цей процес, можна буде поступово знайти найкращі параметри.

2.2.2 Алгоритми визначення епіцентру

Точне розташування будь-якого джерела, що випромінює сейсмічну енергію, є одним із найважливіших завдань практичної сейсмології, і час від часу більшість сейсмологів беруть участь у цьому завданні. У разі землетрусу місце розташування джерела визначається його гіпоцентром (x0, y0, z0) та часом початку t0. Зазвичай гіпоцентр вважається фізичним розташуванням точки початку (або ініціювання) процесу розриву, як правило, задається у довготі (x0), широті (y0) і глибині під поверхнею (z0 [км]). Для простоти гіпоцентр буде позначений x0, y0, z0 з розумінням того, що його можна виміряти в географічних або декартових координатах, тобто в [град] або [км] відповідно. Час початку –

час початку розриву землетрусу. Епіцентр – це проекція гіпоцентру на поверхню Землі (x0, y0). Коли землетрус великий, фізичний розмір зони розриву може становити кілька сотень квадратних кілометрів, і гіпоцентр може бути, в принципі, розташований де завгодно на поверхні розриву, що може мати наслідки і для положення відповідного епіцентру на поверхні Землі.

Гіпоцентр та час початку визначаються часом прибуття, напрямком поширення (обернений азимут) та (або) горизонтальною швидкістю сейсмічних фаз, випромінюваних першими хвилями землетрусу. Це вірно при використанні будь-яких фаз P або S, оскільки їх швидкість поширення завжди більша за швидкість розриву. Отже, енергія P- або S-хвилі, що випромінюється від кінця тривалого розриву, завжди надходитиме пізніше енергії, випромінюваної від її початку. Інформаційні джерела про землетруси (наприклад, від Міжнародного сейсмологічного центру, ISC або Національного інформаційного центру землетрусів (NEIC) USGS) повідомляють про час походження (ЧП) та місцезнаходження, в основному, за часом прибуття високочастотних перших хвиль типу P. Ці параметри ЧП та розташування можуть сильно відрізнятися від часу центроїда та розташування центроїда, отриманих за допомогою інверсії тензора моментів довгоперіодних хвиль, що може бути для справді великих землетрусів навіть більше ніж на хвилину пізніше ЧП і більше ста кілометрів від початкового гіпоцентру. Причина полягає в тому, що ці визначення центроїдів представляють середній час та місце розташування всього сейсмічного моменту, випущеного землетрусом, припускаючи, що останній можна розглядати як точкове джерело у просторі із встановленою симетричною трикутною функцією швидкості моменту. Тим не менше, навіть це припущення не завжди справедливо, особливо для великих багаторазових подій.

2.2.2.1. Один сейсмограф в зоні дії землетрусу

Загалом, епіцентри визначаються за допомогою багатьох спостережень сейсмічної фази з різних сейсмічних станцій. Однак можна також знайти джерело сейсмічного коливання за допомогою однієї 3-компонентної станції.

Оскільки вектор руху хвилі P поляризований у вертикальній площині поширення, його можна розкласти на вертикальну та радіальну складові руху. Це дозволяє розрахувати обернений азимут BAZ до епіцентру (рис. 2.5). Обернений азимут визначається як кут напрямку від станції до епіцентру, виміряний за годинниковою стрілкою з півночі. Радіальну складову P можна реконструювати з записів 2-х горизонтальних компонентів сейсмометра північ-південь та схід-захід, а азимут AZI руху частинок можна розрахувати за співвідношенням амплітуд першого надходження фази P:

$$AZI = \arctan\frac{A_E}{A_N}, \qquad (2.1)$$

де A_E – амплітуда компоненти з напрямком північ-південь;

A_N – амплітуда компоненти з напрямком схід-захід.

Однак формула (2.1) залишає двозначність у 180 градусів, оскільки полярність першого руху у вертикальній складовій може бути вгору або вниз. Щоб отримати правильний обернений азимут, це повинно бути враховано. Тільки якщо перший рух по вертикальній складовій фази P спрямований вниз, відповідний рух горизонтальних частинок відображається до епіцентру, а тоді, коли AZI вимірюється в першому квадранті, BAZ дорівнює AZI. Протилежний випадок, коли перший рух в Z-координаті спрямований вгору. У цьому випадку радіальна складова руху фази P спрямована в бік від гіпоцентра, і, отже, BAZ = AZI + 180 (рис. 2.5).

Згідно з Wiechert [26], справжній кут падіння i_{true} хвилі P є функцією середнього відношення v_P до v_S в корі під сейсмографічною станцією [27].

$$i_{true} = \arcsin\left[\frac{V_P}{V_S} \times \sin\left(0.5 \times i_{app}\right)\right], \qquad (2.2)$$

де v_P – швидкість P-хвилі;

v_S – швидкість S-хвилі;

i_{app} – кут падіння.

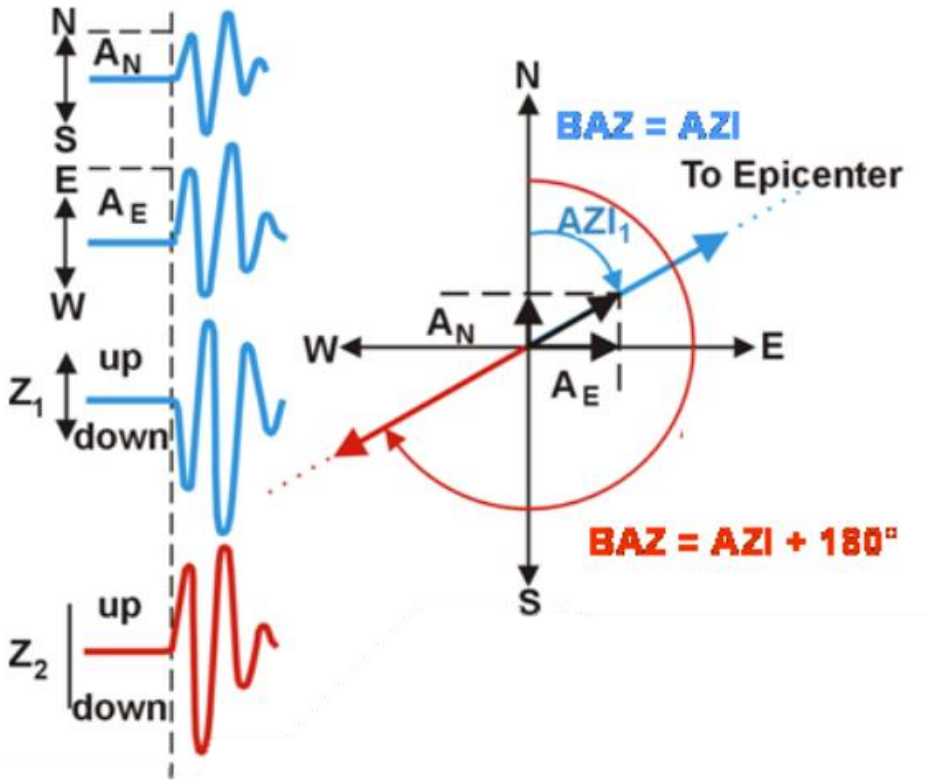

Рисунок 2.5 – Приклад перших рухів хвилі P на 3-компонентних записах (ліворуч), з яких азимут (AZI) (і, отже, обернений азимут (BAZ)) та кут падіння і можуть бути отримані відповідно до рівнянь. (2.1) та (2.2) [25]

З різницею, що враховує амплітудні спотворення внаслідок відбиття перетвореної енергії хвилі P і S на вільній поверхні. Знаючи кут падіння i_o та локальну сейсмічну швидкість v_o під станцією спостереження, ми можемо обчислити видиму горизонтальну швидкість поширення v_{app} цієї сейсмічної фази за допомогою:

$$V_{app} = \frac{V_o}{\sin i_o}, \qquad (2.3)$$

де V_o – локальна сейсмічна швидкість;

i_o – кут падіння.

Однак слід зазначити, що описані тут виміряні значення i_{app} і, таким чином, v_{app} на даній станції залежать від домінуючої частоти сейсмічного запису. Отже, ці значення є репрезентативними для самих верхніх шарів Землі, що відповідають домінуючій довжині хвилі спостережуваної фази, а не для кута

падіння та видимої горизонтальної швидкості безпосередньо під поверхнею Землі.

У випадку повільно виникаючих перших коливань P або при аналізі вузькосмугових високочастотних даних це може бути неможливо правильно прочитати амплітуду та полярність першого. Однак коефіцієнт амплітуди між компонентами повинен залишатися постійним не тільки для першого коливання фази P, але і для наступних коливань тієї ж фази, за умови, що вони ще не спотворені суперпозицією з поперечно заломленою (розсіяною) енергією хвилі , як це часто буває у високочастотних записах. Якщо ця умова виконана, ми можемо, використовуючи цифрові дані, використовувати метод прогнозованої когерентності [28] для автоматичного обчислення оберненого азимута, а також кута падіння. Це набагато надійніше і швидше, ніж використання амплітуд першого руху, що зчитуються вручну, для обчислення обернений азимута з 3-компонентних записів окремих станцій, і це стало звичною практикою [29]. У разі сейсмічних масивів видиму швидкість та обернений азимут можна безпосередньо виміряти, спостерігаючи за розповсюдженням сейсмічного хвильового фронту за допомогою масивних методів, незалежно від місцевих сейсмічних швидкостей під станцією. Як ми показуємо пізніше, спостереження за обернений азимутом корисні для кращого обмеження місць епіцентру та для прив'язки спостережень до сейсмічної події. Знання кута падіння i_o на станції і неявно параметр променя $p = \sin i_o / v_o$ початку на станції допомагає ідентифікувати сейсмічну фазу та розрахувати епіцентральну відстань за допомогою теоретично відомої з відстані повільної повільності (або параметр променя) різних сейсмічних фаз. З однією станцією ми тепер маємо напрямок до сейсмічного джерела. Відстань також можна отримати з різниці часу прибуття двох фаз, як правило, P і S. Якщо прийняти постійну швидкість та час початку t_0, то час P-та S-прибуття можна записати як:

$t_p = t_0 + \frac{D}{V_P}$ та

$t_p = t_0 + \frac{D}{V_S}$, (2.4)

де t_p і t_s – час прибуття P-хвиль та S-хвиль відповідно;

v_p і v_s – швидкості P і S відповідно;

D – епіцентральна відстань для поверхневих джерел; або гіпоцентральна відстань d для глибших джерел.

Виключаючи t0 з рівняння (2.4), відстань можна розрахувати як:

$D = (t_s - t_p)\frac{V_P \times V_S}{V_P - V_S}$. (2.5)

Але формула (2.5) застосовується лише для різниці в часі шляху між Sg і Pg, тобто прямими фазами кори S і P відповідно, за умови, що відношення V_p/V_s не змінюється всередині кори. Pg і Sg є першими коливаннями P і S хвилевих груп місцевих подій лише на відстані приблизно 100 - 250 км, залежно від товщини кори та глибини джерела в корі. Поза цими відстанями фази Pn і Sn, будучи або головними хвилями, критично заломленими при розриві Мохоровичіча [30], або хвилями, що занурюються як хвилі тіла в верхній частині верхньої мантії. Відстань x_{co} між "Pn" та "Pg" (або Pb) може бути приблизно розрахована для значення біля поверхні з залежності:

$x_{co} = 2 \times z_m\{(V_m - \bar{V}_p)(V_m + \bar{V}_p)\}^{-\frac{1}{2}}$, (2.6)

де $\bar{V}_p$– середня швидкість P кори;

V_m – швидкістю P-мохо P;

z_m – товщиною кори.

Вставивши приблизні середні значення $\bar{V}_p$ = 6 км/с та V_m = 8 км/с, ми отримуємо:

$x_{co} \approx 5\ z_m$.

На менших відстанях ми можемо бути впевнені, що спостережений перший поштовх - фази Pg. Однак зауважимо, що це правило діє лише для

поверхневого фокусування. Нижні фази земної кори Pb і Sb тут менш важливі, оскільки вони можуть утворювати лише на дуже короткому епіцентральному діапазоні відстані перші надходження P і S поштовхів. За відсутності локальних кривих часу подорожі для розглянутої області можна використати рівняння (2.5) для виведення "великого правила" для приблизних визначень відстані від різниці часу Sg-Pg у часі. Для ідеального твердого тіл $V_s = V_p/\sqrt{3}$. Це хороше наближення для середніх умов у корі

2.2.2.2. Декілька сейсмографів в зоні дії землетрусу

Коли доступні принаймні 3 сейсмічні станції, можна просто зробити ручне розташування з малювання кіл (метод круга) з центром у місцях розташування станцій та радіусами, рівними гіпоцентральним відстаням, розрахованим за часом S-P (рис. 2.6).

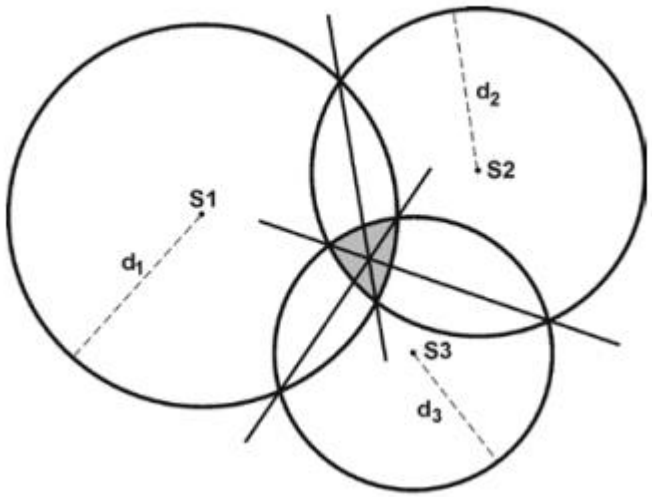

Рисунок 2.6 – Знаходження розташування методом “кола”. Станції розташовані в точках S1, S2 та S3. Епіцентр знаходиться в затіненій зоні, де кола перекриваються. [25]

Ці кола рідко перетинатимуться в одній точці. Це може бути пов'язано з помилками в показаннях часу початку, в передбачуваному часі пробігу або пов'язаної з ними моделі швидкості та (або тим, що ми помилково визначили

поверхневий фокус. Насправді t_s - t_p – це різниця в часі для гіпоцентральної відстані d, яка для сейсмічних джерел із z> 0 км, як правило, більша за епіцентральну відстань Δ (або D). Отже, кола, намальовані навколо станцій радіусом d, як правило, не перетинатимуться в одній точці епіцентру, а скоріше будуть «надмірними». Величина перевищення залежить від глибини джерела, але також від неоднорідностей, що не враховуються нижче середньої швидкості. Тому слід зафіксувати епіцентр або в "центрі ваги" ділянки, що перекривається (темніша зона на рис. 2.6), або намалювати "хорди", тобто прямі лінії, що проходять через точки перетину між двома сусідніми колами. Ці лінії перетинаються в епіцентрі. Ще існують інші методи [31] для вирішення цієї проблеми глибини (наприклад, метод гіперболи, який використовує лише перші надходження хвиль P і передбачає постійну швидкість хвилі P). Детальний опис методу можна знайти у [32]. Протягом останніх років Системи раннього попередження про землетруси (EEWS) стали в центрі уваги, і для таких систем моніторингу потрібні швидкі та надійні методи локалізації землетрусів. Одним із підходів для вирішення проблеми є використання порядку часу першого прибуття для визначення місця події [33] або застосування похідних методу гіперболи [34].

2.2.2.3. Діаграма Вадаті

На основі даних з кількох станцій, доступних з місцевого або регіонального сейсмічного джерела, час виникнення можна визначити за допомогою дуже простої методики, яка називається діаграмою Вадаті (рис. 2.7) [35]). Використовуючи рівняння (2.5) та виключаючи різницю в часі S-P можна розрахувати як

$$t_s - t_p = \left(\frac{V_P}{V_S} - 1\right) \times (t_p - t_0), \qquad (2.7)$$

де t_s – час прибуття S-хвилі;

t_p – час прибуття P-хвилі;

V_P – швидкість P-хвилі;

V_S – швидкість S-хвилі.

Спостережувані часові дані S-P наносять на графік проти абсолютного P-часу. Оскільки t_s - t_p дорівнює нулю в гіпоцентрі, пряма лінія на діаграмі Вадаті (рис. 2.7) дає час початку координат на переході з віссю приходу хвилі P та з нахилу (V_p/V_s - 1) кривої , отримаємо відношення V_p/V_s. Зауважимо, що таким чином можна отримати визначення як часу початку, так і середнього співвідношення V_p /V_s без будь-якого попереднього знання про структуру кори, єдине припущення полягає в тому, що V_p /V_s є постійним, а фази P і S того самого типу, як Pg і Sg, Pb і Sb або Pn і Sn. Таке незалежне визначення цих параметрів може бути дуже корисним при використанні інших методів розташування.

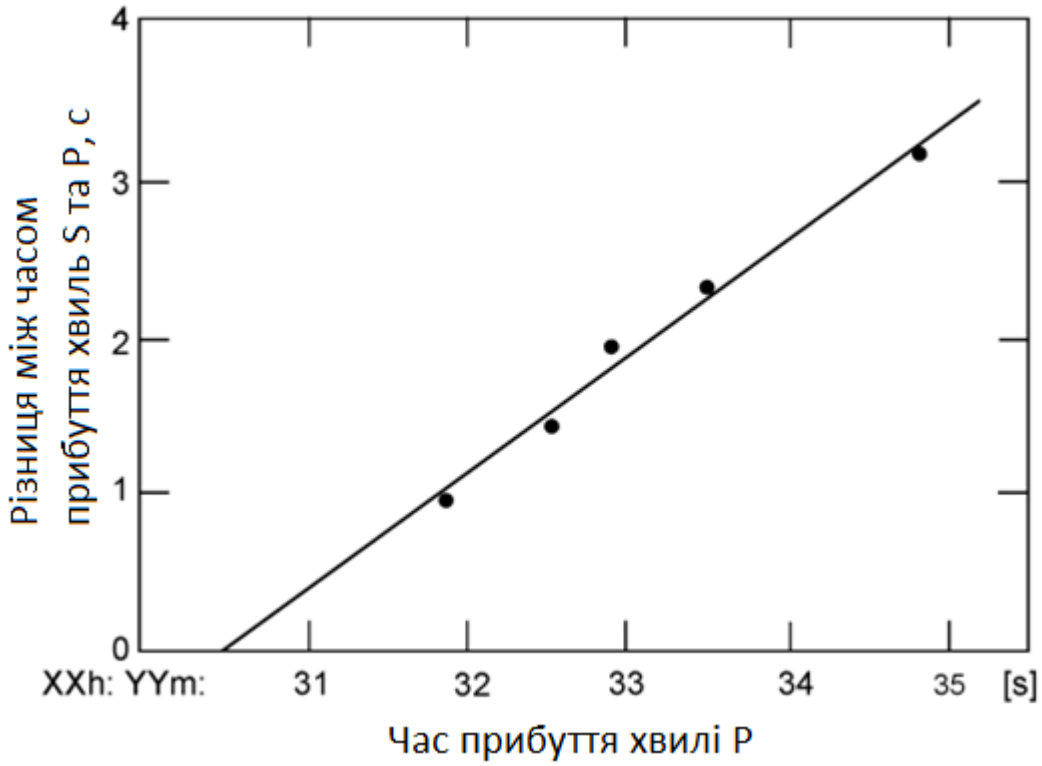

Рисунок 2.7 – Довільний приклад діаграми Вадаті. Перетин апроксимуючої лінії через дані з віссю x дає час початку координат OT. У даному випадку нахил лінії дорівнює 0,72. [25]

Діаграма Вадаті також дуже корисна для незалежних перевірок спостережуваного часу прибуття. Будь-які точки, які недостатньо добре відповідають лінійному співвідношенню, можуть бути погано ідентифіковані,

або через те, що вони не належать до одного фазового типу, або через неправильне зчитування.

2.2.2.4. Оцінка похибки локалізування

Оскільки сейсмічні події визначаються з часом прибуття, який містить помилки спостереження, а час руху обчислюється за неправильним припущенням, що відома точна модель, усі визначення гіпоцентру матимуть помилки. Контур середньоквадратичного пошуку в сітці (рис. 2.8) вказує на невизначеність епіцентру. Так само можна було б зробити тривимірні контури, щоб отримати вказівку на тривимірну невизначеність. Питання в тому, як кількісно оцінити цей показник. Середньоквадратичне значення остаточного рішення дуже часто використовується як критерій „доброти придатності”. Хоча це може бути індикацією, середньоквадратичне значення залежить від кількості станцій і саме по собі не вказує на помилки, а середньоквадратичне значення не повідомляється, наприклад, NEIC та ISC.

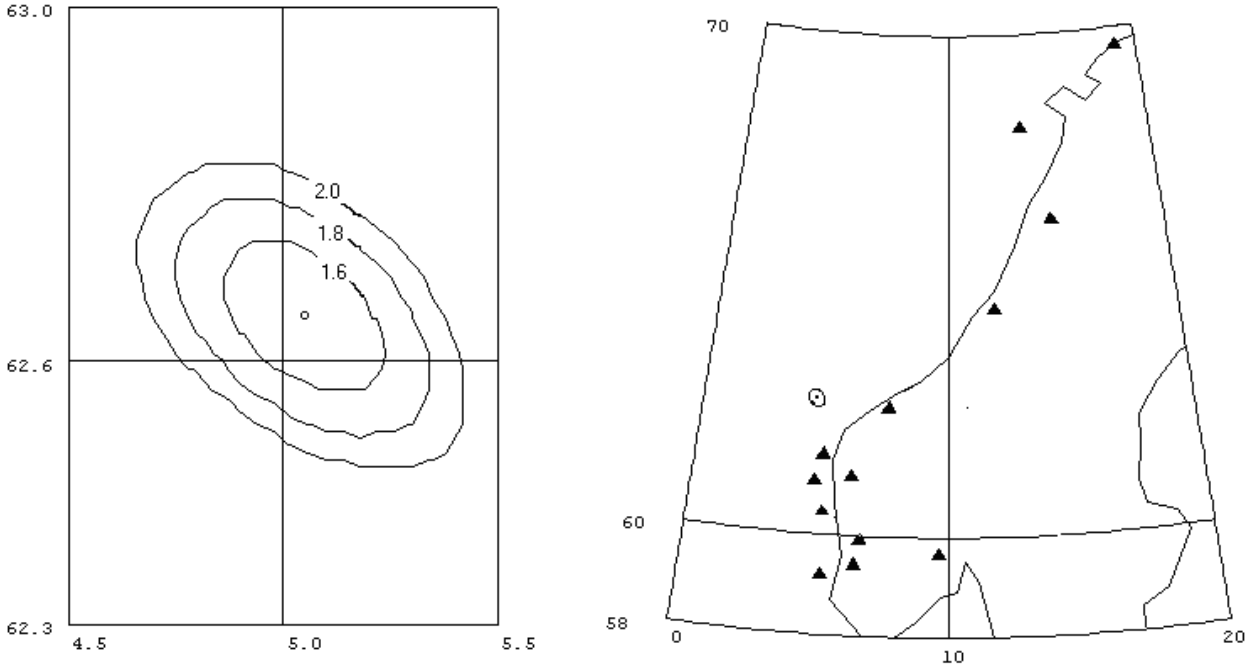

Рисунок 2.8 – Ліворуч: середньоквадратичні контури (у секундах) від місця пошуку землетрусу у західній Норвегії. Розмір сітки становить 2 км. Коло

посередині вказує точку з найнижчий середньоквадратичний показник (1,4 с).

Справа: Місце землетрусу та використовувані станції.

З рис. 2.8 видно, що контури рівних середньоквадратичних значень не є колами. Ми можемо розрахувати контури, в межах яких існує 67% ймовірності (або будь-якої іншої бажаної ймовірності) знаходження епіцентру. Це називається помилковим еліпсом. Так зазвичай відображаються помилки гіпоцентру. Отже, недостатньо вказати одне число для помилки гіпоцентра, оскільки вона просторово змінюється. Стандарти NEIC та ISC дають означення помилкам широти, довготи та глибини, що також може ввести в оману, якщо еліпс помилки не має малої та великої осей NS або EW.

Перш ніж вдаватися до обговорення похибок і помилок, спробуємо скласти розуміння того, які елементи впливають на форму і розмір епіцентрального еліпса помилок. Якщо у нас немає помилок часу прибуття, помилок визначення епіцентру, тому величина помилки (розмір еліпса помилки) повинна бути пов'язана з невизначеністю часу прибуття. Якщо ми припустимо, що всі помилки читання часу прибуття рівні, це може вплинути лише на розмір, а не на форму еліпса помилки.

Зміна геометрії станцій дасть іншу форму еліпса помилок. Таким чином, будь-яка мережа може передбачити форму та орієнтацію еліпсів з помилками, а з урахуванням помилки прибуття, а також розмір еліпса для будь-якого бажаного місця епіцентру. Це можна, наприклад, використовувати для прогнозування того, як зміна конфігурації мережі вплине на місця землетрусів на певній ділянці.

У всіх попередніх тезах передбачалося, що помилки мають гауссовий розподіл і що немає таких систематичних помилок, як помилки годинника і синхронізації часу. Також передбачається, що внаслідок невідомих структур відсутні помилки в теоретичному часі пробігу, обернених азимутах або розрахунках параметрів променя. Це, звичайно, не відповідає дійсності. однак обчислення помилок стає занадто складним, якщо ми не припускаємо простого

розподілу помилок і якщо всі станції мають однакову помилку часу прибуття. Це, однак, означає, що такі розрахунки помилок дають нам лише уявлення про “внутрішню” точність розташувань в рамках моделі, на якій вони базуються. Їх не слід помилково сприймати як справжні помилки розташування.

Попереднє обговорення дало якісний опис помилок. Зараз ми покажемо, як розрахувати фактичні гіпоцентральні помилки з помилок часу прибуття та конфігурації мережі. Найбільш поширений підхід до розташування сейсмічного джерела базується на інверсії найменших квадратів та гауссовому розподілі помилок часу прибуття, і в цьому випадку статистика добре зрозуміла, і ми можемо використовувати розподіл щільності ймовірності χ^2 для обчислення помилок. Для конкретного місцезнаходження χ^2 можна обчислити як:

$$\chi^2 = \frac{1}{\sigma^2}\sum_{i=1}^{n} r_i^2, \qquad (2.8)$$

де σ – передбачуване однакове середньоквадратичне відхилення будь-якого із залишків;

n – кількість спостережень.

Тепер ми можемо розглянути стандартні статистичні таблиці (таблиця 2.1), щоб знайти очікуване значення χ^2 в межах заданої ймовірності. Як видно з таблиці, в межах 5% ймовірності χ^2 - це приблизно кількість ступенів свободи (ndf), яке в нашому випадку дорівнює n-4.

Таблиця 2.1 – Відсотки від розподілення ймовірності χ^2 для різних степенів свободи

ndf	χ^2(95%)	χ^2(50%)	χ^2(5%)
5	1.1	4.4	11.1
10	3.9	9.3	18.3
20	10.9	19.3	31.4
50	34.8	49.3	67.5
100	77.9	99.3	124.3

Якщо, наприклад, подія розташована на 24 станціях (ndf = 20), існує лише 5% ймовірності, що χ^2 перевищить 31,4. Значення χ^2 зростатиме, коли ми будемо віддалятися від найкраще прилягаючого епіцентру, а у наведеному вище прикладі контур, в межах якого χ^2 менше 31,4, відображатиме еліпс помилок, в межах якого є 95% шансів знайти епіцентр. На практиці помилки здебільшого повідомляються з імовірністю 67%, що означає ± 1 стандартне відхилення. Помилки в гіпоцентрі та часі початку також можна формально визначити за допомогою матриці дисперсії – коваріації σ^2_x параметрів гіпоцентралі. Ця матриця визначається як:

$$\sigma_\chi^2 = \begin{Bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 & \sigma_{xz}^2 & \sigma_{xt}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 & \sigma_{yz}^2 & \sigma_{yt}^2 \\ \sigma_{zx}^2 & \sigma_{zy}^2 & \sigma_{zz}^2 & \sigma_{zt}^2 \\ \sigma_{tx}^2 & \sigma_{ty}^2 & \sigma_{tz}^2 & \sigma_{tt}^2 \end{Bmatrix}, \quad (2.9)$$

де елементи діагоналі – це дисперсії параметрів розташування x, y, z і t0. Тоді як елементи, що не мають діагоналі, дають зв'язок між помилками в різних гіпоцентральних параметрах. Докладніше див., Наприклад, Stein (1991) [36]. Приємною властивістю σ^2_x є те, що його легко обчислити:

$$\sigma_\chi^2 = \sigma^2 \times (G^T G)^{-1}, \quad (2.10)$$

де σ^2 – дисперсія часу прибуття, помножена на ідентифікаційну матрицю;

G^T – транспонована матриця G.

Таким чином, стандартні відхилення гіпоцентральних параметрів задаються квадратним коренем діагональних елементів, і це звичайні помилки, про які повідомляється. Оскільки σ^2_x є симетричною матрицею, її може представляти діагональна матриця в системі координат, яка повернена відносно системи відліку. Зараз ми маємо лише помилки в гіпоцентральних параметрах, а еліпс з помилками просто має піввісі σ_{xx}, σ_{yy} та σ_{zz}. Таким чином, основна інтерпретація недіагональних елементів полягає в тому, що вони визначають

орієнтацію та форму еліпса помилок. Тому повне визначення вимагає 6 елементів. Формули (2.9) та (2.10) також показують, що форма та орієнтація еліпса з помилками залежить лише від геометрії мережі та структури кори, тоді як стандартне відхилення спостережень є масштабним фактором, доки вони рівні для всіх спостережень.

Тому критичною змінною в аналізі помилок є дисперсія часу прибуття σ^2. Це значення, як правило, більше, ніж можна було б очікувати лише від помилок синхронізації та вибору, проте воно може відрізнятися від випадку до випадку. Встановлення фіксованого значення для даного набору даних може призвести до нереальних обчислень помилок. Тому більшість програм визначення місцезнаходження визначатимуть σ за залишками найбільш підходящого гіпоцентра:

$$\sigma^2 = \frac{1}{ndf}\sum_{i=1}^{n} r_i^2, \tag{2.11}$$

де ndf – кількість ступенів свободи.

Ділення на ndf, а не на n, компенсує поліпшення придатності в результаті використання часу прибуття з даних. Однак це працює лише частково, і деякі програми дозволяють встановлювати апріорне значення, яке використовується лише в тому випадку, якщо кількість спостережень невелика. Для малих мереж це може бути критичним параметром. Деякі дослідження [37] показали, як для регіональних, так і для місцевих сейсмічних мереж, що оцінка похибки ERH (у горизонтальному напрямку) та ERZ (у вертикальному напрямку), як зазначено за звичайними програмами [38] не можна розглядати як консервативну оцінку справжньої помилки розташування та може призвести до невиправданих тектонічних висновків.

2.3 Обґрунтування вибору створення програмного продукту та використання хмарних технологій для реалізації IoT-рішень.

2.3.1 Використання Google cloud platform

Так як розроблювана система повинна мати можливість до масштабування (розширення регіону моніторингу, додавання інших аналізуючих алгоритмів на основі зібраних даних), найкращим рішенням є розмістити центр обробки і аналізу у хмарі.

Після запуску проекту IoT пристрої сейсмографів будуть виробляти багато даних. Потрібен ефективний, масштабований, доступний спосіб як керувати цими пристроями, так і обробляти всю цю інформацію та змушувати її працювати. Що стосується зберігання, обробки та аналізу даних, особливо великих даних, важко придумати щось краще за хмарне рішення.

На рис. 2.9 показані різні етапи управління даними IoT у Google Cloud.

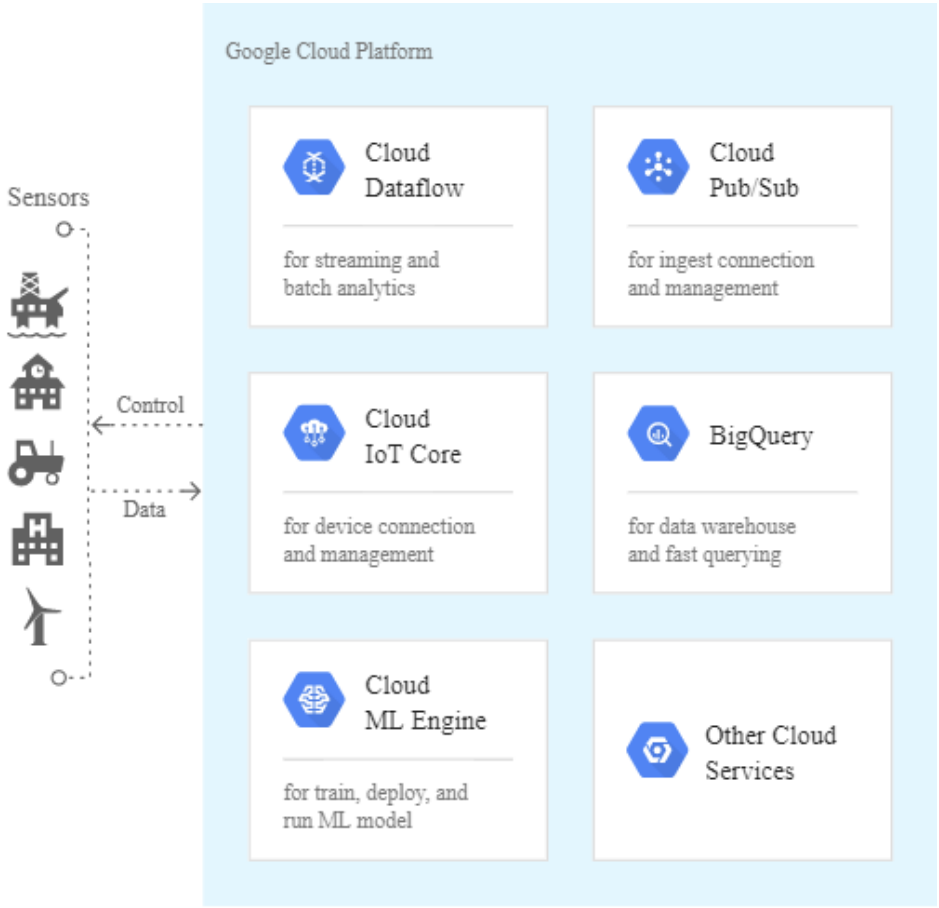

Рисунок 2.9 Хмарна платформа Google Cloud Platform [39]

IoT Core забезпечує повністю керовану послугу для управління пристроями. Це включає реєстрацію, автентифікацію та авторизацію в ієрархії ресурсів Google Cloud, а також метадані пристрою, що зберігаються в хмарі, і можливість надсилати конфігурацію пристрою.

Поглинання (Ingestion) – це процес імпорту інформації з пристроїв у сервіси Google Cloud. Google Cloud надає різні послуги поглинання, залежно від того, чи є дані телеметричними чи оперативною інформацією про пристрої та інфраструктуру IoT.

IoT Core забезпечує безпечну підтримку MQTT. Цей ефективний галузевий стандарт дозволяє обмеженим за ресурсом пристроям надсилати телеметрію в режимі реального часу, а також негайно отримувати повідомлення, надіслані з хмари на пристрій, за допомогою функції управління конфігурацією. Брокер IoT Core MQTT безпосередньо підключається до Pub/Sub.

Pub/Sub надає довготривалу послугу передачі повідомлень. Створюючи теми для потоків або каналів, можна дозволити різним компонентам додатку підписуватися на певні потоки даних без необхідності створювати канали для кожного абонента на кожному пристрої. Pub/Sub також безпосередньо підключається до інших сервісів Google Cloud, допомагаючи підключити систему передачі даних, конвеєри даних та системи зберігання.

Багато пристроїв мають обмежені можливості зберігати та повторно відправляти дані телеметрії. Pub / Sub масштабує для обробки стрибків даних, які можуть виникати, коли багато пристроїв реагують на події у фізичному світі, та буферизує ці піки, щоб допомогти ізолювати їх від програм, що контролюють дані.

Конвеєри керують даними після їх надходження на Google Cloud, подібно до того, як керовано деталями на заводській лінії. Сюди входять такі завдання, як:

- **Перетворення даних.** Ви можете перетворити дані в інший формат, наприклад, перетворивши захоплену напругу сигналу пристрою в калібровану одиницю виміру температури.
- **Об'єднання даних та обчислень.** Комбінуючи дані, ви можете додавати перевірки, такі як усереднення даних на декількох пристроях, щоб уникнути дії на одному, помилковому пристрої. Ви можете переконатися, що у вас є активні дані, якщо один пристрій переходить у автономний режим. Додавши обчислення до конвеєра, ви можете застосувати потокову аналітику до даних, поки вони все ще знаходяться в конвеєрі обробки.
- **Збагачення даних.** Ви можете поєднувати генеровані пристроєм дані з іншими метаданими про пристрій або з іншими наборами даних, такими як погода або дані про дорожній рух, для використання в подальшому аналізі.
- **Переміщення даних.** Ви можете зберігати оброблені дані в одному або кількох місцях остаточного зберігання.

Потік даних забезпечує відкриту модель програмування Apache Beam як керовану послугу для обробки даних різними способами, включаючи пакетні операції, шаблони вилучення-перетворення-навантаження (ETL) та безперервні потокові обчислення. Потік даних може бути особливо корисним для управління конвеєрами обробки великих обсягів даних, необхідних для сценаріїв IoT. Потік даних також призначений для безперебійної інтеграції з іншими сервісами Google Cloud, які ви вибрали для свого конвеєра.

Дані з фізичного світу надходять у різних формах та розмірах. Google Cloud пропонує цілий ряд рішень для зберігання даних: від неструктурованих великих масивів даних, таких як зображення або відеопотоки, до структурованого об’єкта зберігання пристроїв чи транзакцій та високоефективних реляційних баз даних для даних про події та телеметрії.

Враховуючи, що пристрої IoT можуть деякий час проводити в режимі сну з низьким енергоспоживанням і можуть існувати в особливо ненадійних мережах, часто корисно зберігати частину стану пристрою в хмарі. Таким чином, дані стану можуть бути доступними навіть тоді, коли самі пристрої тимчасово перебувають у режимі офлайн.

Останній відомий стан пристрою можна повідомляти та зберігати в IoT Core для отримання додатками. Інформація про стан, надіслана через MQTT або HTTP, зберігається в IoT Core і доступна в хмарі, навіть якщо пристрій відключено або відключено.

2.3.2 Зберігання даних додатків у Datastore та Firebase

Коли вам потрібно зробити дані стану або телеметрії доступними для мобільних або веб-додатків, ви можете зберігати оброблені або необроблені дані у структурованих, але без схем базах даних, таких як Datastore та Firebase Realtime Database, де ви можете представляти дані пристрою IoT як об'єкти домену або рівня програми .

Аналіз правил обробки та потокової передачі даних у хмарних функціях та потоці даних.

Події та дані IoT можуть надсилатися до хмари з високою швидкістю, і їх потрібно швидко обробити. Для багатьох додатків IoT рішення розмістити пристрій у фізичному середовищі приймається з метою забезпечення швидшого доступу до даних. Наприклад, продукція, яка під час транспортування зазнає високих температур, може бути негайно позначена та утилізована.

Вміння швидко обробляти та діяти з цією інформацією є ключовим. Cloud Functions дозволяє писати власну логіку, яка може застосовуватися до кожної події в міру її надходження. Це може бути використано для активації сповіщень, фільтрації недійсних даних або виклику інших API. Хмарні функції можуть працювати з кожною опублікованою подією окремо.

Якщо вам потрібно обробляти дані та події за допомогою більш досконалої аналітики, включаючи методи віконного перегляду або збігу даних із декількох

потоків, Dataflow надає високопродуктивний інструмент аналітики, який можна застосувати до потокових та пакетних даних.

Тепер, щоб вставити дані у BigQuery, будемо користуватися хмарними функціями Firebase, які можна налаштувати на виконання на основі багатьох різних тригерів та подій. Одним із таких активаторів є нові дані, вставлені в топік PubSub, тому ми будемо слухати наш топік, пов’язану з нашим Реєстром пристроїв, і з кожними даними що надходять, ми виконуємо функцію, яка зберігає дані у BigQuery та підтримує останні дані пристрою у Firebase Realtime.

У нашій системі за допомогою Firebase, будемо надсилати push-повідомлення користувачам, на основі даних про наявність землетрусів та локації епіцентру з кластеру Google cloud platform.

2.4 Висновки після другого розділу

У розділі розглянуто вимоги до проекту IoT-рішення. Наведено методи детекції землетрусу, що будуть використовуватись в датчиках-сейсмографах. Наведено алгоритми пошуку епіцентру землетрусу, що будуть частиною хмарної інфраструктури центру обробки та аналізу. Проаналізовано хмарне рішення Google Cloud Platfotm та спеціалізований сервіс IoT core, та інтеграцію Google Firebase.

Розділ 3. Реалізація архітектури IoT-рішення

3.1 Архітектура проекту IoT-рішення.

Сейсмографи передають дані про коливання до центру обробки і аналізу, у ролі якого виступає сервер. Сервер має бути розрахований на підключення до сотні клієнтів одночасно. При виявленні початку землетрусу сейсмограф має надіслати дані до центру обробки і аналізу. Центр обробки і аналізу визначає гіпоцентр і епіцентр землетрусу, та надсилає попередження жителям. Запропоновану архітектуру наведено на рис. 3.1.

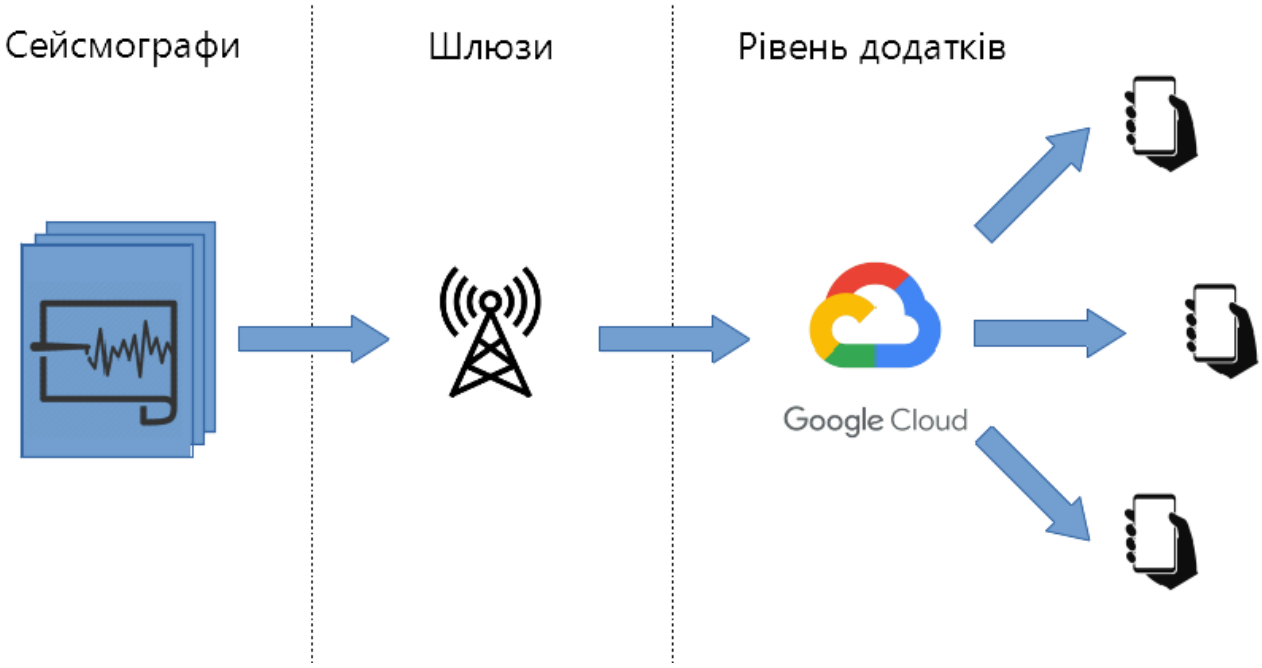

Рисунок 3.1 – Запропонована архітектура IoT рішення

Згідно технічного завдання, у розробці потрібно використовувати передові прийоми виготовлення деталей і складання, сучасні програмовані технології.

3.2 Комунікаційні технології та системи.

Проаналізувавши аналоги і зробивши деякі висновки, було вирішено використовувати технологію мобільного зв'язку. На щастя, в Закарпатському регіоні (на час написання данної роботи), вже багато де існує покриття 3G та 4G. На рис. 3.2 зображено карту покриття мобільного зв'язку третього та четвертого покоління найбільшими мобільними операторами України [40].

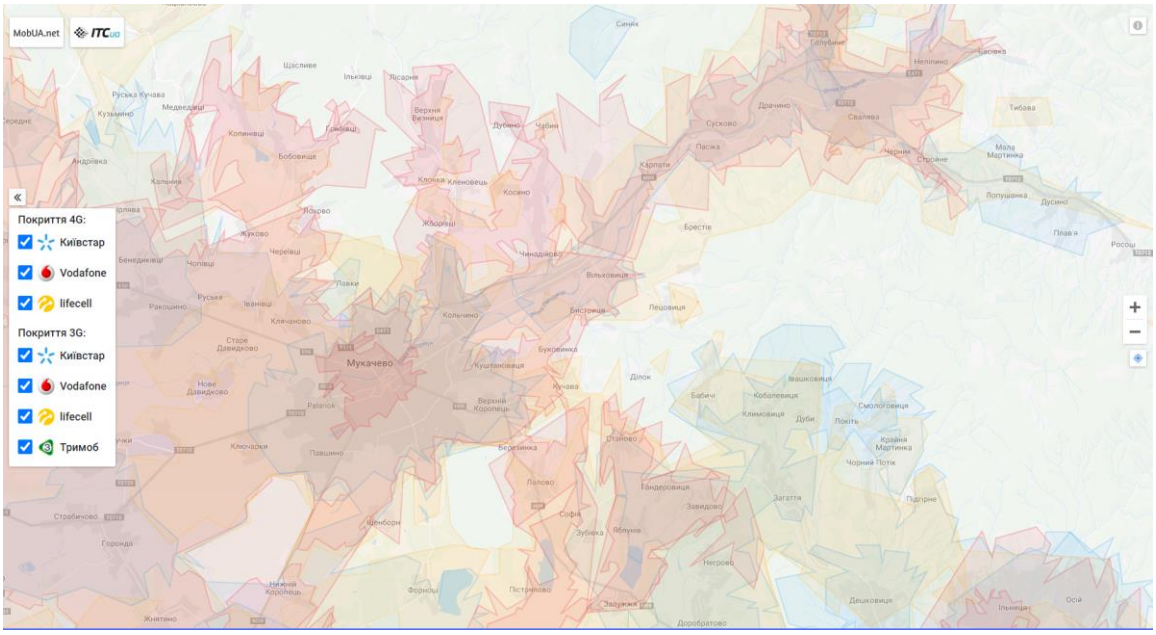

Рисунок 3.2 – Покриття 3G та 4G на Закарпатті

Вважаю, що таке рішення позбавить недоліків, що були виявлені аналізуючи аналогічні рішення.

Щодо протоколу прикладного рівня, вважаю, що найкраще підійде MQTT. Це дуже легкий мережевий протокол, що є надбудовою над протоколом TCP/IP. Використовується для обміну повідомленнями між пристроями за принципом видавець-підписник (рис. 3.3) [41].

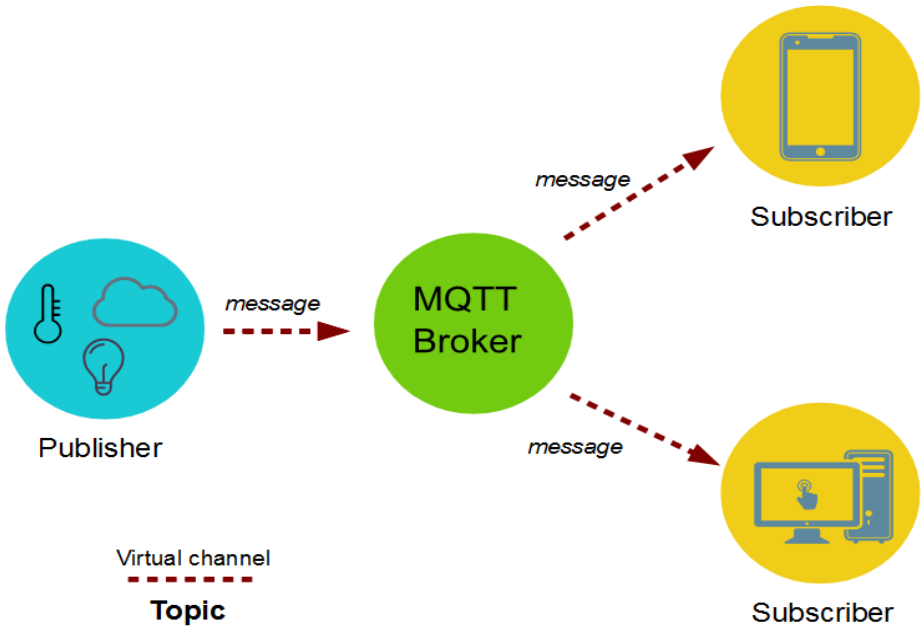

Рисунок 3.3 – Протокол MQTT [41]

Відкритий, простий стандарт, розроблений так, щоб його було легко імплементувати. Ці характеристики роблять його ідеальним протоколом для використання в багатьох ситуаціях, наприклад, для спілкування в контекстах Machine - Machine (M2M) та Internet of Things (IoT), де потрібен невеликий розмір коду та(або) невелика пропускна здатність мережі.

До особливостей MQTT належать:

1 Використання схеми повідомлень видавець/підписник, яка забезпечує розподіл повідомлень від одного до багатьох, та розподілення програм.

2 Транспортний потік для обміну повідомленнями, який є агностичним щодо вмісту корисного навантаження, який він доставляє.

3 Три рівня захисту з огляду доставки повідомлень:

1) «Найбільше відразу», повідомлення доставляються відповідно до максимальних зусиль операційного середовища, де може статися втрата повідомлення. Цей рівень може використовуватися, наприклад, з даними датчиків навколишнього середовища, де не має значення, чи буде втрачено індивідуальне зчитування, оскільки наступне буде опубліковано незабаром після.

2) "Принаймні один раз", де повідомлення надійдуть, але можуть виникати дублікати.

3) "Рівно один раз", де запевняється, що повідомлення надійде рівно один раз. Цей рівень може бути використаний, наприклад, для платіжних систем, де повторювані або втрачені повідомлення можуть призвести до неправильних платежів.

4 Мінімальна частка службових транспортних кадрів для зменшення мережевого трафіку.

5 Механізм сповіщення зацікавлених сторін про аномальне відключення.

3.3 Схемотехнічне проектування

3.3.1 Розробка структурної схеми та принцип роботи модулю

В технічному завданні викладені основні функціональні компоненти пристрою. Функціональна структура пристрою складається з наступних блоків:

- Мікроконтролер
- Акселерометри
- Мобільний трасміттер

Структура сейсмографічного пристрою зображена на рис. 3.4.

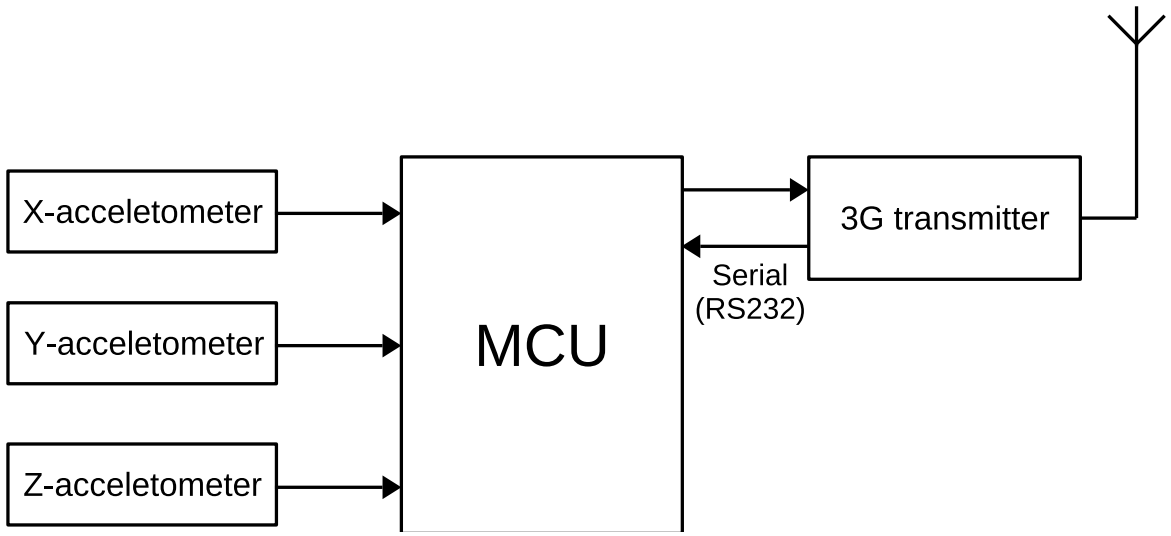

Рисунок 3.4 – Блок-схема пристрою сейсмографу

Акселерометри детектують землетрус, міряючи прискорення по трьом осям. Після подолання порогу спрацювання, мікросхема акселерометра генерує переривання, що виводить мікроконтролер із режиму очікування. Мікроконтролер записує один кадр коливань, та «пакує» дані для транспортування по протоколу MQTT за допомогою 3G трансміттера, що з'єднується за допомогою послідовного порту.

3.3.2 Вибір та обґрунтування елементної бази

У цьому розділі буде наведено технічне та економічне обгрунтування вибору елементної бази для датчика-сейсмографа. Зауважимо, що 3G/4G модуль,

що буде використовуватись – SIM7600E-H, це є вимогою ТЗ. Готова схема знаходиться на креслені ІРм21.402221.001Е3.

3.3.2.1. Вибір мікроконтролеру

При виборі мікроконтролеру, потрібно чітко розуміти, яку кількість пам’яті буде займати вбудована програма. Було вирішено обрати мікроконтролер з «запасом» та з тієї серії, на яку є багато документації та власного досвіду роботи з нею. Таким мікроконтролером став STM32F401RET6. Надалі, коли весь необхідний функціонал буде імплементовано, можна обрати менш продуктивний мікроконтролер, щоб не переплачувати за невикористані потужності.

3.3.2.2. Вибір акселерометру

Одна з найголовніших складових пристрою, акселерометр, нам потрібно вибрати з урахуванням вимог точності та надійності. Якісні датчики дають хороші результати, оскільки вони мають великий коефіцієнт SNR (сигнал/шум). Однак через їхню високу ціну, їх важко використовувати у великих масштабах для виявлення землетрусів, тому що це економічно невигідно. Тому має бути а компроміс між якістю та вартістю. Ключові параметри кожного датчика розглянуті нижче. Датчик прискорення фіксує рухи точніше, коли рівень шуму низький і чутливість висока. Виходячи з цього факту, я оцінив чотири датчики прискорення ADXL355 [42], LIS3DHH [43], MPU9250 [44] та MMA8452 [45], вивчивши їх технічні характеристики. Таблиця 2.1 підсумовує критерії оцінки (параметри) чотирьох датчиків. Чутливість – це вплив зміни найменшого біта. Ширина виходу пов’язана з чутливістю, тобто це вихідна роздільна здатність. Максимальна швидкість даних – найбільша вихідна частота даних.

Як результат, згідно параметрів, наведених у таблиці 3.1, ADXL355 найкращий сенсор, оскільки рівень шуму низький, а рівень чутливості найвищий із усіх чотирьох датчиків. LIS3DHH – другий кращий у списку. Хоча його чутливість не вище за MPU9250, рівень шуму набагато нижче, ніж у MPU9250. Обидві критерії оцінки ADXL355 та LIS3DHH кращі, ніж інші два датчики.

Оскільки нашою метою є розробка пристрою виявлення землетрусів на основі IoT, нам потрібно вибрати відповідний датчик прискорення, який відповідає вимогам точності та низької ціни. Тому виходячи з параметрів, оберемо LIS3DHH як сейсмічний датчик, оскільки його точність вище порівняно до MPU9250 та MMA8452, а вартість порівняно низька (приблизно 10 доларів США за мікросхему) порівняно з ADXL355 ($50 США).

Таблиця 3.1 – Параметри IC акселерометрів

Серія IC	Параметри					
	Шум $(^{g}/_{\sqrt{Hz}})$	Чутливість $(^{mg}/_{bit})$	Ширина виходу (біт)	Масштаб (g)	Максимальна швидкість даних (Hz)	Ціна ($)
ADXL355	25	0,0039	20	2,048	1000	50
LIS3DHH	45	0,076	16	2,5	1100	10
MMA8452	126	9,76	12	2	800	3
MPU9250	300	0,061	16	2	4000	3

Як результат, розглядаючи як точність, так і ціну, LIS3DHH є найбільш підходящим для нашого пристрою сейсмографа.

3.3.2.3. Вибір мікросхеми контролеру заряду акумулятора

Пристрій є портативним, та має у своїй конструкції акумулятор, тому до складу схеми необхідно включити ланцюг заряду li-po акумулятора. Було обрано три універсальних мікросхеми для заряду li-po та li-ion акумуляторів: Texas Instruments BQ21040DBVR [46], Microchip MCP73812T [47] та NanJing Top Power TP4056 [48] (таблиця 3.2).

Таблиця 3.2 – Параметри IC контролерів заряду акумулятору

Серія IC	Параметри				
	Максимальний струм заряду (А)	Похибка напруги заряду (%)	Можливість підключення термістору	Діапазон робочих температур (°С)	Ціна ($)
MCP73812T	0,5	±1	-	-40 - 80	0,46
TP4056	1	±1,5	+	-40 - 85	0,05
BQ21040	0,8	±1	+	0 - 125	1
Ваговий коеф. b_j	0,35	0,15	0,25	0,1	0,15

Вагові коефіцієнти підібрані у відповдності до важливості конкретного параметру, більш важливому параметру відповідає більший коефіцієнт. Складаємо матрицю X відповідно до таблиці основних параметрів, прибираємо проміжки і вагові коефіцієнти (таблиця 3.3).

Таблиця 3.3 – Матриця приведених параметрів

X =	0,5	1	0	120	0,46
	1	1,5	1	125	0,05
	0,8	1	1	125	1

Тепер потрібно привести матрицю приведених параметрів до матриці |N| - нормованих параметрів, користуючись формулою:

$$a_{ij} = \frac{(\max\limits_{j} x_{ij} - x_{ij})}{x_{ij}}, \quad (2.1)$$

де: $\max\limits_{j} x_{ij}$ – максимальне значення в стовпчику j матриці X;

x_{ij} – поточне значення в стовпчику j матриці X.

Матриця нормованих параметрів |N| наведена у таблиці 3.4.

Таблиця 3.4 – Матриця нормованих параметрів

N =	MCP73812T	0,5	0	1	0,04	0,891
	TP4056	0	0,333	0	0	0
	BQ21040DBVR	0,2	0	0	0	0,95

Введемо оціночну функцію Q_i, яка визначається формулою:

$$Q = \sum_{j=1}^{n} a_j b_j, \tag{2.2}$$

де: a_j – нормований параметр у стовпчику j;

b_j – ваговий коефіцієнт для стовпчика j.

Таблиця 3.5 – Значення оціночних функцій для кожної мікросхеми

Назва	Значення оціночної ф-ції
MCP73812T	0,429
TP4056	0,05
BQ21040DBVR	0,07

Визначивши Q_i для кожної мікросхеми, обираємо ту, яка найкраще задовольняє вимоги ТЗ. Найменшому значенню Q_i відповідає краща мікросхема. В цьому випадку, мікросхема TP4056 підходить найкраще.

3.3.2.4. Вибір мікросхеми стабілізації напруги

Було обрано три мікросхеми, які можуть забезпечити функціональні модулі пристрою стабільною напругою 3,3 В: Texas Instruments TPS63001 [49], TPS613221 [50] та Recom Power RPM3.3-1.0 [51].

Таблиця 3.6 – Параметри ІС стабілізаторів напруги

Серія ІС	Параметри			
	Діапазон вхідних напруг (В)	Максимальний вихідний струм (А)	Діапазон робочих температур (°С)	Ціна ($)
TPS63001	1,8 - 5,5	0,8	-40 - 85	1
TPS613221	0,9 - 5,5	0,75	-40 - 85	0,92
RPM3.3-1.0	3 - 5	1	-40 - 105	4,2
Ваговий коеф. b_j	0,35	0,25	0,15	0,25

Після видалення діапазонів, нормування матриці та множення кожної характеристики на відповідний ваговий коефіцієнт, отримали значення оціночних функцій для кожної мікросхеми .

Таблиця 3.7 – Значення оціночних функцій для стабілізаторів напруги

Назва	Значення оціночної ф-ції
TPS63001	0,1213
TPS613221ADBVR	0,153
RPM3.3-1.0	0,1952

Вибір зроблено на користь TPS63001 від Texas Instruments.

3.3.3 Розробка та розрахунок схеми електричної принципової

3.3.3.1. Ланцюг заряду акумулятора

Схема сейсмографу живиться від li-po акумулятору, що підключається до роз’єму XS2. 5В постійної напруги живлення для заряду акумулятору поступає з роз’єму micro USB female XS1.

Використовуваний у проекті МК STM32F401RET6 живиться напругою 3,3 В. Напруга на одній Li-po батареї, в залежності від рівня її заряду, може коливатися від 3 В (повністю розряджена, спрацьовує захист від перерозряду контролером всередині батареї, що відсікає навантаження), до 4.2 В (повністю заряджена) [52]. Для правильного заряду таких батарей використовуються контролери заряду. У схемі було використано TP4056 від NanJing Top Power для заряду li-ion, li-po батарей, максимальний струм заряду задається резистором R_{prog} (R_3 у схемі), і це значення може бути у діапазоні від 0,45 А до 1 А. Було обрано струм 1 А, так як планується використання акумулятору, ємність якого рівна, або перевищує 2 $А * год$, а загальне правило для заряду li-po, li-ion виглядає так [53]:

$$\max I_{\text{заряду}} = \frac{1}{2} C \,, \tag{3.1}$$

де: С – ємність акумулятору.

Схема заряду акумулятору зображена на рис. 3.5.

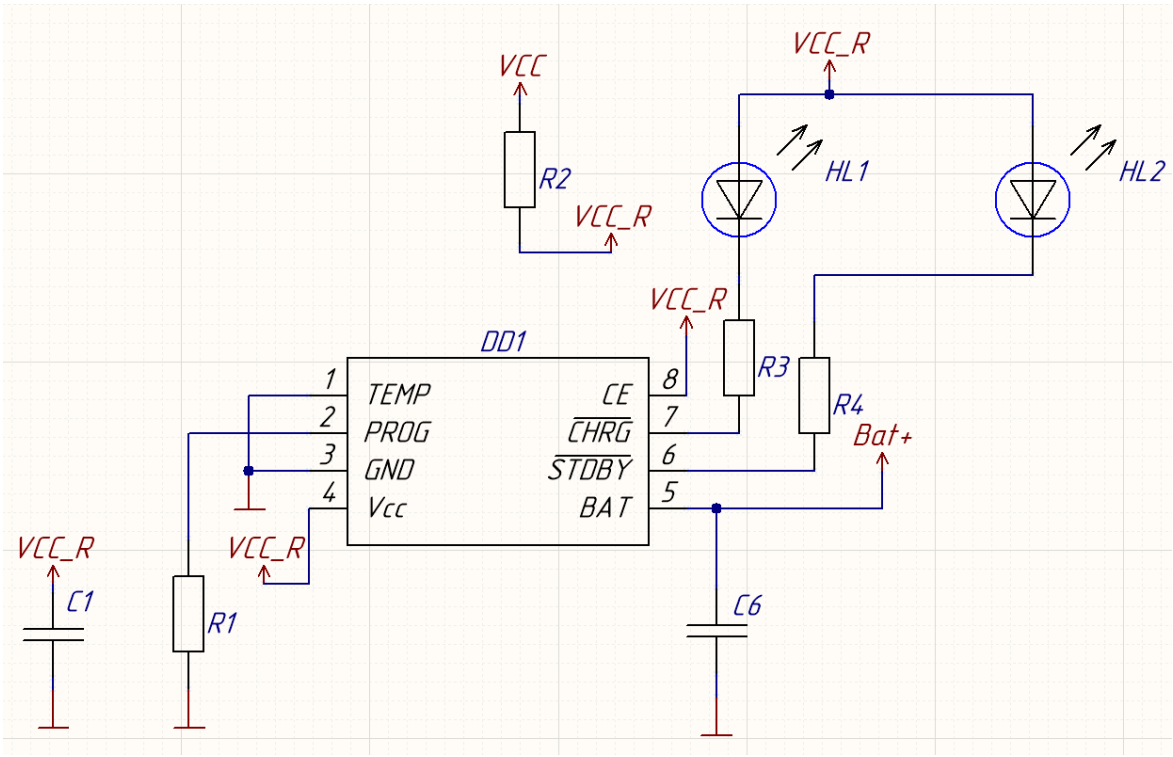

Рисунок 3.5 – Схема заряду акумулятора

На цій схемі бачимо два світлодіоди, HL1 – червоний, індикує про те, що відбувається процес зарядки акумулятору, HL2 – зелений, індикує про те, що процес заряджання завершено.

3.3.3.2. Ланцюг стабілізації

Для стабілізації напруги з акумулятору, та отримання стабільної напруги 3,3 В використовується мікросхема TPS63001. Її підключення обрано згідно рекомендаціям технічної документації на дану мікросхему [49]. На вході живлення мікросхеми ставляться керамічні конденсатои номіналом 10 мкФ та 0,1 мкФ. Таке рішення прийнято з досвіду та зважуючи на технічну документацію. Такі конденсатори називаються блокуючими та використовуються для мінімізації сплесків напруги живлення. Схема стабілізації напруги 3,3 В зображена на рис. 3.6.

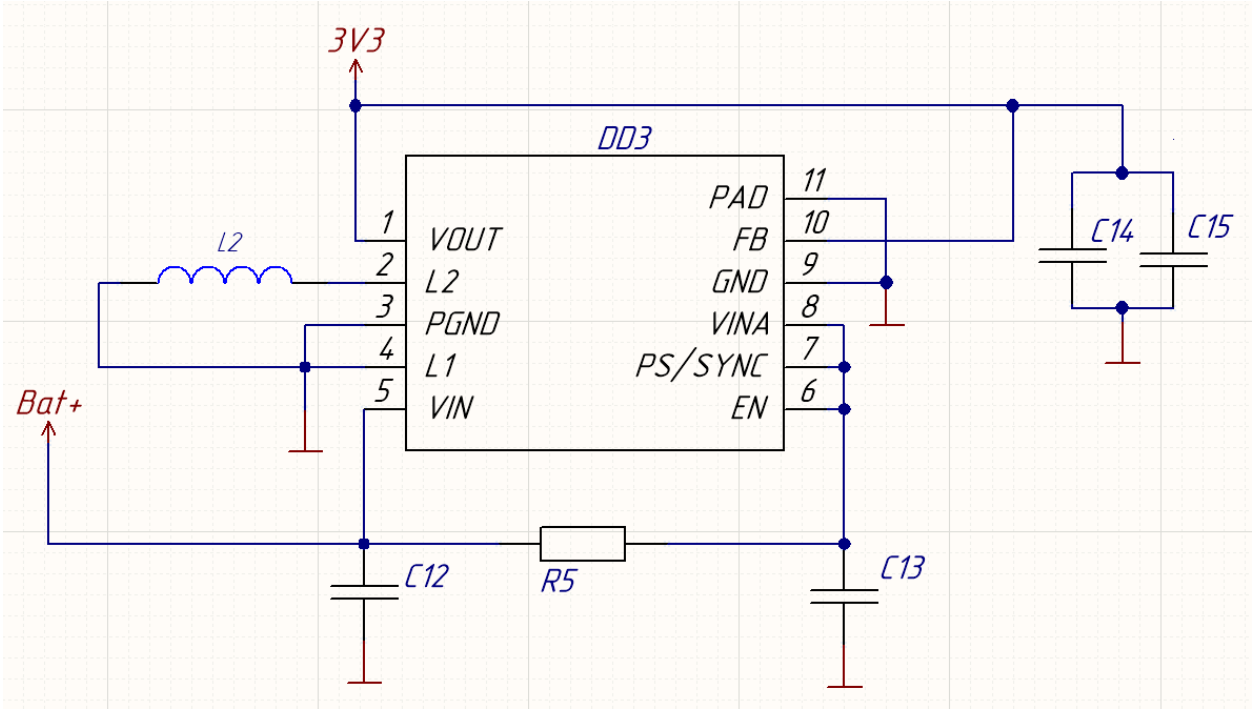

Рисунок 3.6 – Схема стабілізації напруги з акумулятору

Цей ланцюг може працювати з вхідним діапазоном напруг 1,8-5,5В даючи на виході стабільні 3,3В.

3.4 Висновки після третього розділу.

У третьому розділі розглянуто комунікаційні протоколи. Наведено архітектуру системи. Виконано схемотехнічне проектування пристрою сейсмографа. Обгрунтовано вибір елементної бази, зокрема, акселерометру, мікроконтролеру. Розроблено та розраховано схему електричну принципову.

Розділ 4. Програмна реалізація інтерфейсних рішень, програмування мікроконтролерів та реалізація проекту, використання розподілених систем та хмарних технологій

4.1 Проектування програмного забезпечення для мікропроцесору

Для того, щоб система працювала, нам потрібно написати програму, яка при появі поштовхів записувала б інтенсивність цих поштовхів по трьом осям, та надсилала по 3G/4G модулю на центр обробки і аналізу за допомогою MQTT (рис. 4.1).

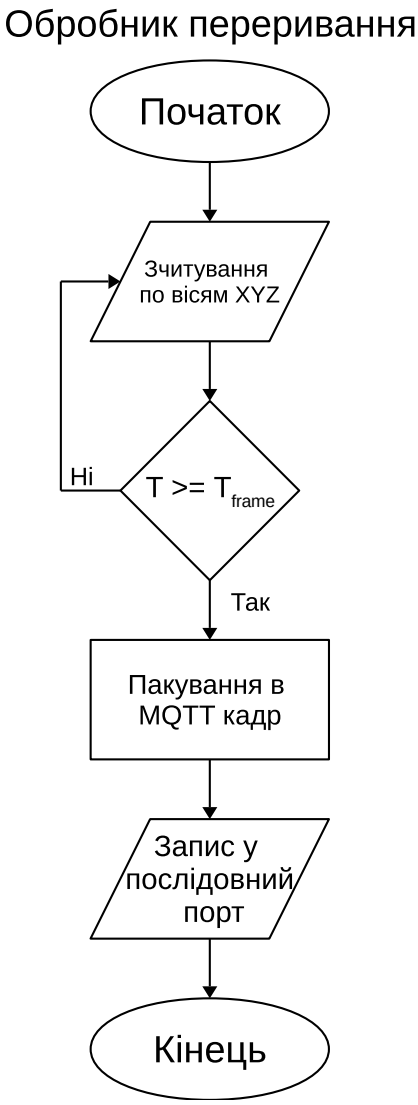

Рисунок 4.1 – Обробник переривання у мікроконтролері

Нормальний режим роботи мікроконтролеру – режим очікування. В цьому режимі споживана потужність найменша. Як тільки переривання генерується акселерометром, мікроконтролер «прокидається» і виконує код у обробнику переривань, а саме, зчитує значення прискорень протягом часу T_{frame}, що є константою, та формує MQTT пакет для надсилання через 3G трансміттер. Код програми наведено у додатках.

4.2 Проектування додатку для центра обробки і аналізу

У прототипі IoT системи для моніторингу сейсмічної активності, центр обробки і аналізу поєднує в собі функціонал MQTT брокеру, та виконує аналітику даних. На рис. 4.2 зображено архітектуру хмарного додатку центру обробки і аналізу, створеної на базі хмарної інфраструктури Google cloud platform.

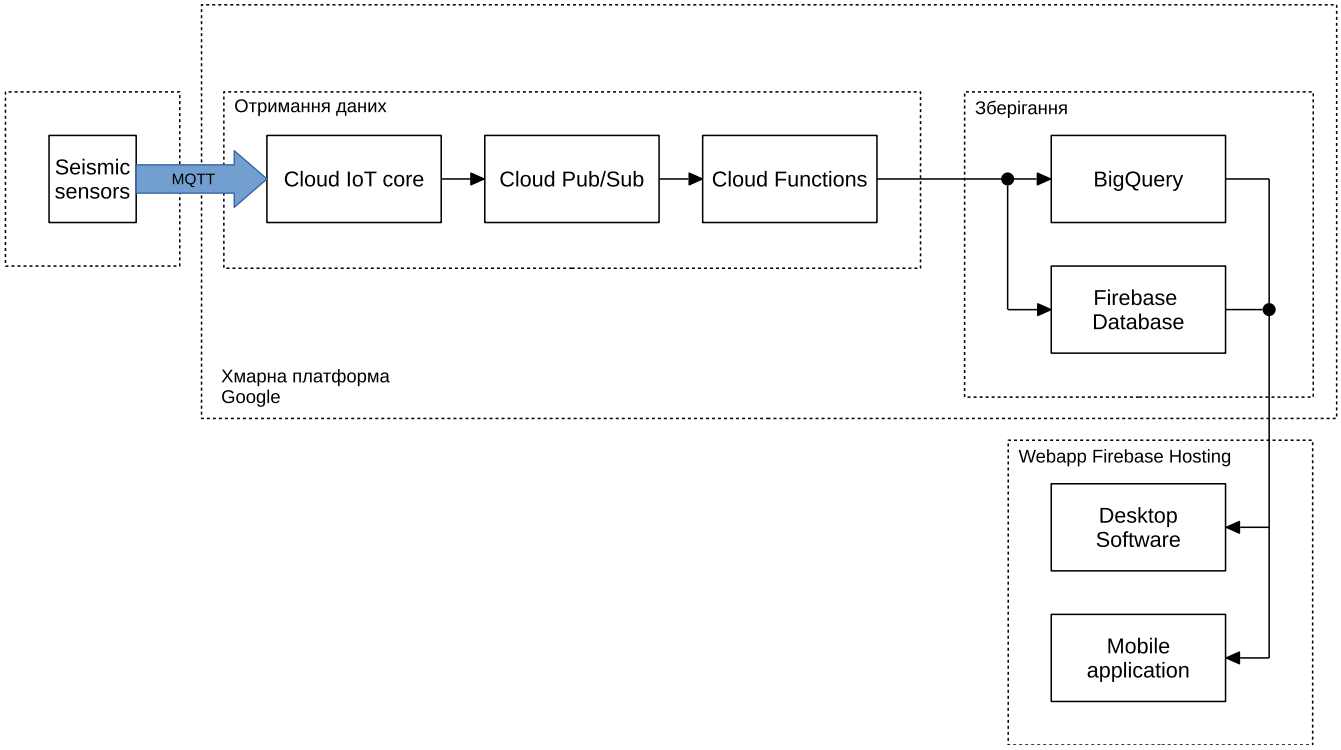

Рисунок 4.2– Архітектура хмарного додатку на платформі Google

На стороні Cloud IoT Core налаштовано Cloud PubSub – один з основних компонентів для процесингу вхідних даних. Робота з цим компонентом заключалась в наступних діях:

- Створення дозволу для IoT Core для постингу повідомлень у PubSub компоненті.
- Створення MQTT топіку, у який і будуть відправлятися повідомлення.
- Створення підписок для читання MQTT топіку
- Додавання так званого регістру через який прилади-сейсмографи будуть зареєстровані і зможуть підключатись до Cloud IoT Core

Кожний сейсмограф тепер можна зареєструвати використовуючи дані створеного нами регістру. Після цього дані будуть надсилатись у обраний топік. Використовуючи BigQuery (теж сервіс Google), будемо зберігати дані з сенсорів. Firebase Realtime Database корисна для зберігання даних в реальному часі, тобто телеметричні показники. Також слугує для синхронізації між підключеними клієнтами (в нашому випадку смартфони жителів регіону та Windows додаток для перегляду історичних даних).

У контейнері **Cloud Functions** можна створити обробники івентів. У нас такими івентами слугуватимуть пороги спрацювання пристроїв-сейсмографів. Тобто, коли дані надсилатимуться трактом MQTT, обробник івентів підхопить їх. Обробник івентів викликає методи, що слугують для аналізу зчитаних коливань, та локалізують епіцентр коливань. На основі цих даних приймається рішення для пересилання повідомлень смартфонам у зоні дії, за допомогою Firebase Realtime Database.

4.3 Проектування графічного інтерфейсу користувача

Додаток для перегляду історичних даних реалізовано у вигляді Windows програми. На рис. 4.3 зображено головний екран програми у режимі перегляду історичних даних. На формі бачимо вікно для перегляду історичних даних, поля для вказання діапазону дат для перегляду та перемикач режимів відображення – кількість землетрусів по дням або кількість землетрусів відповідної інтенсивності у цьому періоді.

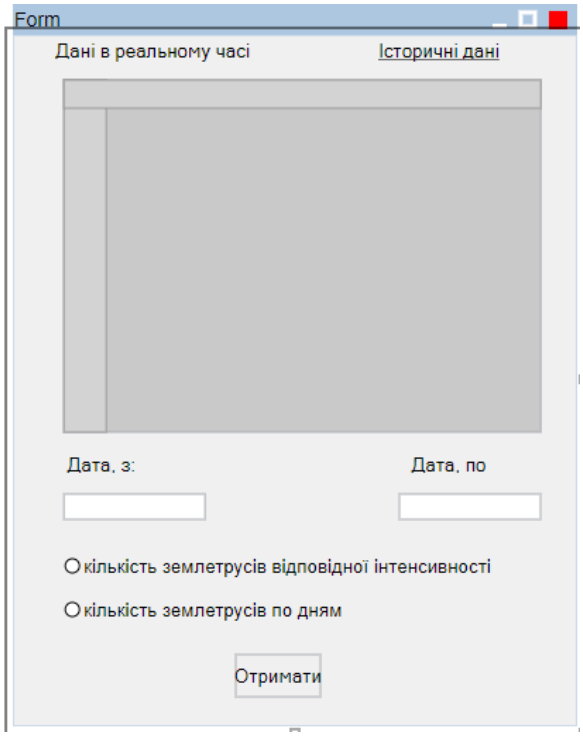

Рисунок 4.3 – Вікно перегляду історичних даних

Програма отримує дані через запити до хмарної бази даних Firebase. Код програми наведено у специфікації.

4.4 Проектування мобільного додатку

За допомогою конструктору застосунків Appery [54], було створено MVP нативного додатку для Android під назвою «**EarthShake**».

Додаток розроблявся як доповнення до інфраструктури системи IoT моніторингу та попередження про землетруси. Функціонал альфа-версії включає прийом push повідомлень про землетрус, для користувачів що знаходяться у зоні дії землетрусу. Для цього використовується геолокація мобільного пристрою. Користувач має змогу налаштувати поріг спрацювання push-повідомлень, налаштувавши:

- магнітуду землетрусу,
- мінімальну відстань від теоретичного епіцентру коливань.

Також є вікно перегляду історичних даних землетрусів що відбувались у даному регіоні та вікно перегляду коливань у реальному часі. Конструктор

Appery дозволяє інтегрувати бекенд push-повідомлень, та використовувати їхню хмарну базу данних. На рис. 4.4 ліворуч зображено головний екран застосунку, де можна переглянути історичні дані землетрусів, праворуч – меню застосунку.

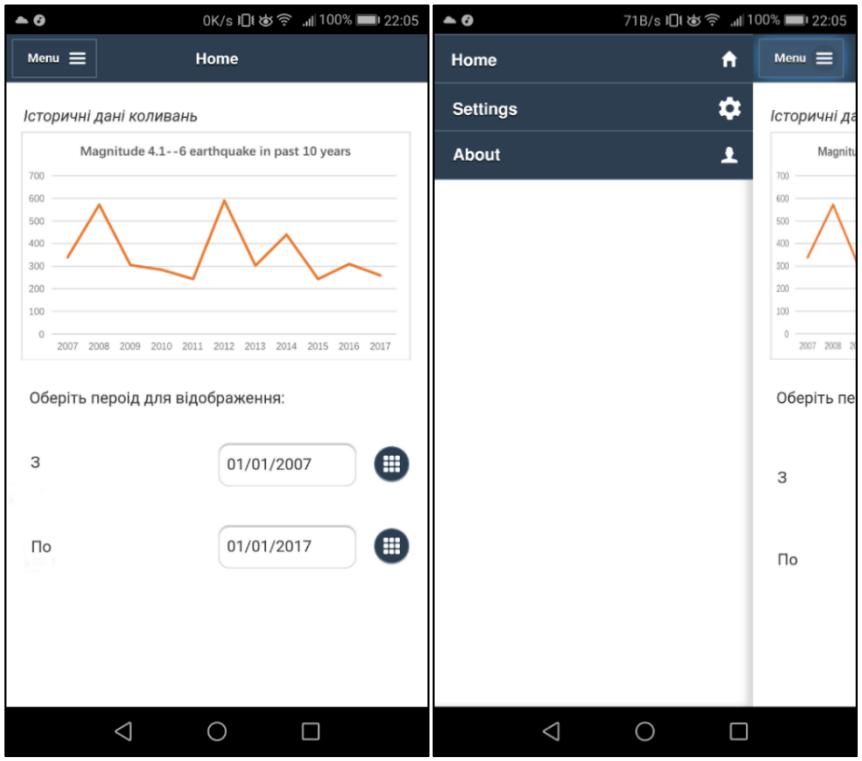

Рисунок 4.4 – Головний екран (ліворуч) та меню застосунку

Меню застосунку містить 3 пункти: головний екран, екран налаштувань, про додаток.

Меню налаштувань (рис. 4.5) дозволяє увімкнути сповіщення про землетруси, встановити два пороги спрацювання сповіщень:

- відстань від епіцентру у кілометрах;
- поріг інтенсивності землетрусу (шкала за Ріхтером [55]).

Вмикання сповіщень здійснюється повзунком «Попередження про землетруси».

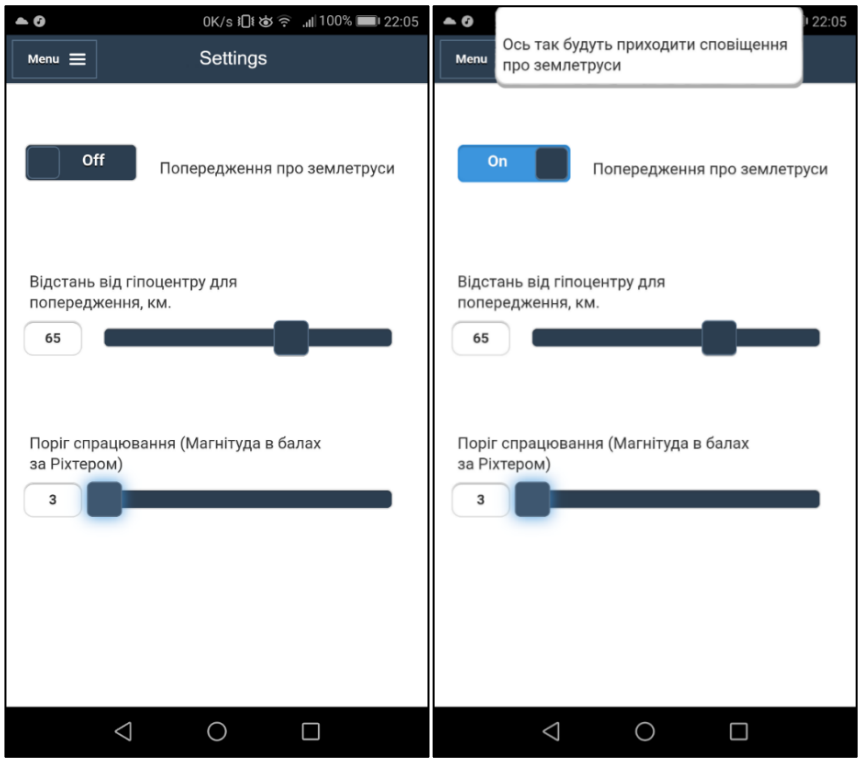

Рисунок 4.5 –Меню налаштувань

Сповіщення приходитимуть якщо смартфон знаходиться у заданому радіусі від епіцентру (гіпоцентру) землетрусу.

4.5 Висновки після четвертого розділу

У розділі розглянуто програмну частину екосистеми IoT для моніторингу сейсмологічної ситуації в регіоні. Описано роботу прошивки для мікроконтролеру STM32. Наведено архітектуру хмарного додатку для центру обробки та аналізу. Описано роботу графічного додатку для перегляду історичних даних на ОС Windows. Продемонстровано MVP додатку для ОС Android для попередження жителів регіону.

ВИСНОВКИ

У рамках кваліфікаційної роботи магістра було проаналізовано існуючі рішення та алгоритми, що дозволяють покращити точність детектування землетрусів. Відповідно до мети проекту, якою було проектування та практична реалізація системи IoT рішень для моніторингу сейсмологічної ситуації на прикладі Закарпатського регіону, яка відрізняється від аналогічних рішень простотою алгоритму роботи, мінімальною затратою енергоресурсів мікроконтролера, доступністю по апаратно-програмних технологіях та ціні.

Було виконано наступні завдання:

- проведено аналіз існуючих систем інформування про стихійні лиха;
- виконано дослідження технології зв’язку для даної системи;
- сформовано уявлення про конструктивні та структурні рішення;
- описано логічні зв’язки вузлів системи на етапі проектування;
- наведено структурну схему роботи системи IoT для моніторингу та інформування про землетруси;
- виконано верифікацію даних та апробацію результатів.

Новизною даного проекту стало покращення алгоритмів детектування землетрусів та отпимізація їх для використання у дешевих низькопотужних мікроконтролерах. Також до новизни можна віднести використання новітньої елементної бази, зокрема, проектування і використання алгоритмів для безперебійної передачі даних у високонавантажених системах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kravchenko O., Sakharov D. IoT seismological situation monitoring system development with one of the regions of Ukraine as an example //Information Technology and Interactions (Satellite): Conference Proceedings, December 04, 2020, Kyiv, Ukraine / Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor). Kyiv: Stylos, 2020. Pp. 373 ISBN 978-966-2399-61-5

2. Кравченко О.В., Сахаров Д.Ю. Створення IoT системи моніторингу сейсмологічної ситуації на прикладі одного з регіонів України// XVI міжнародна науково-технічна конференція «ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ», 2021 р.

3. Кравченко О.В., Сахаров Д.Ю. Створення IoT системи моніторингу сейсмологічної ситуації на прикладі одного з регіонів України// VI міжнародна науково-практична конференція «Інформаційні технології та взаємодії», 2019 р., ст. 252.

1. What Is the Internet of Things (IoT) [Електронний ресурс] // Oracle. – 2019. – Режим доступу: http://bit.ly/2qadArR. (дата звертання 21.03.2021)

2. Сейсмічність України. Дані Інституту геофізики НАН України ім. С.І. Субботіна. [Електронний ресурс] // СЦД-Україна. – 2019. – Режим доступу: http://bit.ly/37W71di. (дата звертання 15.05.2020)

3. Nicoletta Brazzola, Simon Helander. Five approaches to build functional early warning systems. // UNDP Europe and the CIS Istanbul Regional Hub, Pp 16-23.

4. Alphonsa A., Ravi G. «Earthquake Early Warning System by IOT using Wireless Sensor Networks» // 2016 International Conference on Wireless Communications, Signal Processing and Networking *(матеріали конференції)*

5. Jangsoo Lee, Irshad Khan, Seonhwa Choi, Young-Woo Kwon «A Smart IoT Device for Detecting and Responding to Earthquakes» [Електронний ресурс] // Electronics 2019 – Режим доступу: https://bit.ly/2ABoUm5 (дата звертання 15.05.2020)

6. Bishop, C. Neural Networks for Pattern Recognition; Oxford University Press: New York, NY, USA, 1996.

7. A.M. Zambrano, I. Perez, C. Palau, M. Esteve. Technologies of Internet of Things applied to an Earthquake Early Warning System // Future Generation Computer Systems (2016).

8. Geopsy project. STA/LTA algorithm. [Електронний ресурс] – Режим доступу до ресурсу: http://www.geopsy.org/wiki/index.php/Geopsy:_STA/LTA (дата звертання 02.04.2021)

9. Amadej Trnkoczy. Understanding and parameter setting of STA/LTA trigger algorithm // Information Sheet. University of Potsdam (1999)

10. Hüseyin Yigitler, Behnam Badihi and Riku Jäntti. Overview of Time Synchronization for IoT Deployments: Clock Discipline Algorithms and Protocols. // Sensors (ISSN 1424-8220; CODEN: SENSC9).

11. Oracle. Internet gateway. [Електронний ресурс] – Режим доступу до ресурсу:https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/managingIGs.htm (дата звертання 02.04.2021)

12. Rajalakshmi Krishnamurthi, Adarsh Kumar, Dhanalekshmi Gopinathan , Anand Nayyar, Basit Qureshi. An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques. // Sensors (ISSN 1424-8220; CODEN: SENSC9).

13. Chuyang Gao, Shuai Zhao, Bo Cheng. Design and Implementation of Real Time and History multi-view IoT trend Display and Control System // 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). ISBN:978-1-5386-8179-4.

14. Ravi Sharma, Shiva Prakash, Pankaj Roy. Methodology, Applications, and Challenges of WSN-IoT. // 2020 International Conference on Electrical and Electronics Engineering (ICE3). ISBN:978-1-7281-5847-1.

15. Johana A. Manrique, Johan S. Rueda-Rueda, Jesús M.T. Portocarrero. Contrasting Internet of Things and Wireless Sensor Network from a Conceptual Overview. // 2016 IEEE International Conference on Internet of Things (iThings) and

IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). ISBN:978-1-5090-5881-5.

16. C. P. Kruger, G. P. Hancke. Implementing the Internet of Things vision in industrial wireless sensor networks. // 2014 12th IEEE International Conference on Industrial Informatics (INDIN). Electronic ISBN:978-1-4799-4905-2.

17. M. Hussein M. Zorkany, Neamat S. Abdel Kader. Design and Implementation of IoT Platform for Real Time Systems. // The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018). AMLTA 2018. Advances in Intelligent Systems and Computing, vol 723. Springer, Cham.

18. ScienceDirect. Network Time Protocol. [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/computer-science/network-time-protocol (дата звертання 02.04.2021)

19. Digital guide Ionos. Simple network time protocol: the stripped-back protocol for time synchronization. [Електронний ресурс] – Режим доступу до ресурсу: https://www.ionos.com/digitalguide/server/know-how/sntp-simple-network-time-protocol/

20. Texas Instruments. AN-1728 IEEE 1588 Precision Time Protocol Time Synchronization Performance. // Application Report SNLA098A–October 2007–Revised April 2013.

21. UPSeis. What Are Seismic Waves? [Електронний ресурс] – Режим доступу до ресурсу: http://www.geo.mtu.edu/UPSeis/waves.html

22. Jens Havskov, Peter Bormann, Johannes Schweitzer. Seismic source location // Information Sheet. University of Potsdam.

23. Beno Gutenberg. The Structure of the Earth. // International Geophysics, Academic Press, Volume 1, 1959, Pages 13-20, ISBN 9780123106506.

24. A. Manyele, S. Mwalembe. Seismic Station. // International Conference on Advances in Engineering and Technology, 2006.

25. Hu, Jing, Yu, Haiying. Accurate determination of P-wave back azimuth and slowness parameters by sparsity constrained seismic array analysis. // 2018 Geophysical Journal International.

26. Saari, J. Automated phase picker and source location algorithm for local distances using a single three-component seismic station // (1991) Tectonophysics, 189, Pp. 307-315.

27. Earthquake Glossary. Moho. [Електронний ресурс] – Режим доступу до ресурсу: https://earthquake.usgs.gov/learn/glossary/?term=Moho

28. M. Bath. Introduction to Seismology. // 1979 Birkhäuser Basel, ISBN978-3-0348-5283-8.

29. J. Pujol. Earthquake location tutorial: graphical approach and approximate epicentral location techniques. // Seism. Res. Lett., 75, 63-74.

30. Nicholson, T., Gudmundsson, Ó., Sambridge, M. Constraints on earthquake epicentres independent of seismic velocity models. // (2004). Geophys. J. Int., 156, Pp. 648-654.

31. Satriano, C., Lomax, A., and Zollo, A. Real-time evolutionary earthquake location for seismic early warning. // (2008) Bull. Seism. Soc. Amer., Pp. 98, 1482-1494

32. Wadati, K. On the travel time of earthquake waves. // Part. II. Geophys. Mag. 1933, Tokyo, Pp-7, 101-111.

33. Stein, S. , Introduction to seismology, earthquakes and Earth structure // 1991, Lecture notes, Department of Geological sciences, Northwestern University, USA.

34. Di Giovambattista, R., and Barba, S. // 1997, An estimate of hypocenter location accuracy in a large network, possible implication for tectonic studies in Italy. Geophys. J. Int., 129, 124-132.

35. Lahr, J. C., W. K., Kanamori, H., Jennings, P. C. // Kisslinger. (Eds.) (2003), 1617-1618

36. Google. Google cloud IoT core. [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/iot-core.

37. Зведена карта покриття всіх 4G/3G операторів України, мобільний 4G та 3G інтернет. [Електронний ресурс] // mobua.net – Режим доступу: https://www.mobua.net/maps/?pos=48.491821,22.930829,10 (дата звертання 15.05.2020)

38. MQTT Protocol Tutorial: Technical description [Електронний ресурс] // Survivingwithandroid. – 2019. – Режим доступу: http://bit.ly/2Y9iSQL (дата звертання 15.05.2020)

39. Low Noise, Low Drift, Low Power, 3-Axis MEMS Accelerometers. [Електронний ресурс] – Режим доступу: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl354_355.pdf (дата звертання 15.05.2020)

40. MEMS motion sensor: three-axis digital output accelerometer. [Електронний ресурс] – Режим доступу: https://www.st.com/resource/en/datasheet/lis3dhh.pdf (дата звертання 15.05.2020)

41. MPU-9250 Product Specification. [Електронний ресурс] – Режим доступу: https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf (дата звертання 15.05.2020)

42. MMA8452Q, 3-axis, 12-bit/8-bit digital accelerometer.[Електронний ресурс] – Режим доступу: https://www.nxp.com/docs/en/data-sheet/MMA8452Q.pdf (дата звертання 15.05.2020)

43. BQ21040 0.8-A, Single-Input, Single Cell Li-Ion and Li-Pol Battery Charger. Datasheet. [Електронний ресурс] – Режим доступу до ресурсу: http://bit.ly/2WZJ4PM. (дата звертання 29.05.2020)

44. MCP73811/2. Simple, Miniature Single-Cell, Fully Integrated Li-Ion/Li-Polymer Charge Management Controllers. Datasheet. [Електронний ресурс] – Режим доступу до ресурсу: http://bit.ly/2I7SXCJ. (дата звертання 29.05.2020)

45. TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8. Datasheet. [Електронний ресурс] – Режим доступу до ресурсу: http://bit.ly/2I6ao6w. (дата звертання 29.05.2020)

46. TPS6300x High-Efficient Single Inductor Buck-Boost Converter With 1.8-A Switches. Datasheet. [Електронний ресурс] – Режим доступу до ресурсу: http://bit.ly/2WwGeNM. (дата звертання 29.05.2020)

47. TPS61322 6.5-μA Quiescent current, 1.8-A switch current boost converter. Datasheet. [Електронний ресурс] – Режим доступу до ресурсу: http://bit.ly/2ME4LRi. (дата звертання 29.05.2020)

48. RPM-1.0 DC/DC Converter. Datasheet. [Електронний ресурс] – Режим доступу до ресурсу: http://bit.ly/2IwsayV. (дата звертання 29.05.2020)

49. Li-Ion & LiPoly Batteries. All about the power packs that propel your projects. [Електронний ресурс] – Режим доступу до ресурсу: http://bit.ly/2wLAhSC. (дата звертання 29.05.2020)

50. BU-409: Charging Lithium-ion. [Електронний ресурс] – Режим доступу до ресурсу: http://bit.ly/2ZeJlM8. (дата звертання 29.05.2020)

51. Appery.io: MBaaS - Enterprise Mobile Application Development [Електронний ресурс] – Режим доступу до ресурсу: https://appery.io (дата звертання 29.05.2020)

52. Richter scale. Definition and facts. [Електронний ресурс] – Режим доступу до ресурсу: https://www.britannica.com/science/Richter-scale (дата звертання 29.05.2020)

Додаток А

СТВОРЕННЯ ІОТ СИСТЕМИ МОНІТОРИНГУ СЕЙСМОЛОГІЧНОЇ СИТУАЦІЇ НА ПРИКЛАДІ ОДНОГО З РЕГІОНІВ УКРАЇНИ

Технічне завдання

Листів 2

Розробник _____ Дмитро САХАРОВ

Н

Київ, 2021

1 Найменування та галузь використання

IOT система для моніторингу сейсмологічної ситуації Закарпатського регіону. Використовується для раннього попередження жителів та туристів регоіну про землетруси.

2 Підстава для розробки

Підставою для розробки є завдання на курсовий проект у другому семестрі першого курсу освітньо-кваліфікаційного рівня «Магістр», спеціальності 126 «Інформаційні системи та технології», освітня програма «Програмні технології інтернет речей».

3 Мета і призначення розробки

Розробка датчику-сейсмографу для детектування землетрусів, передачі даних на центр обробки та аналізу. Система сповіщень жителів та туристів Закарпатського регіону.

4 Технічні вимоги

Склад виробу й вимоги до системи, що розробляється.

Система складається з датчиків-сейсмографів для, власне, детектування землетрусів, та центр обробки і аналізу. Сейсмографи містять у собі сенсори достатньої чутливості щоб реєструвати механічні коливання по двом або трьом осям. Сейсмографи передають дані про коливання до центру обробки і аналізу, у ролі яого виступає сервер. Сервер має бути розрахований на підключення до сотні клієнтів одночасно. При виявленні початку землетрусу сейсмограф має надіслати дані до центру обробки і аналізу. Центр обробки і аналізу визначає гіпоцентр і епіцентр землетрусу, та надсилає попередження жителям.

Вимоги до технологічності.

Орієнтовані на передові прийоми виготовлення деталей і складання.

Вимоги до рівня уніфікації й стандартизації.

Для виготовлення пристрою сейсмографа застосувати стандартні, уніфіковані деталі та вироби.

Умови експлуатації.

Кліматичне виконання сейсмографу УХЛ1.1 по ГОСТ 15150-69.

Додаткові технічні вимоги.

Додаткові технічні характеристики наведені в таблиці А.1:

Таблиця А.1 – Додаткові технічні характеристики.

Температурний режим	-15°C ...+40°C
Вологість	0 % ... 99 %
Живлення	5 В ± 5 % (для заряду)
Струм споживання	не більше 300 мА у активному режимі.

- Пристрій сейсмографу має бути портативним та живитися від однієї li-po або li-ion батареї. Забезпечити автономність потрібно на декілька місяців, тому дуже важливою є вимога малого активного струму споживання.
- У якості 3G/4G модуля використати SIM7600E-H від waveshare.
- Заряджання акумулятору – від роз'єму micro USB.

Додаток Б

СТВОРЕННЯ ІОТ СИСТЕМИ МОНІТОРИНГУ СЕЙСМОЛОГІЧНОЇ СИТУАЦІЇ НА ПРИКЛАДІ ОДНОГО З РЕГІОНІВ УКРАЇНИ

Текст програми для МК STM32

Листів 4

Розробник _____ Дмитро САХАРОВ

Н

Київ, 2021

```
#include <iostream>
#include <unistd.h>
#include <sstream>
#include <ctime>
#include <signal.h>
#include <mqtt.h>
using namespace std;

#include "kits.h"
#if INTEL_IOT_KIT == DFROBOTKIT
#include "dfrobotkit.hpp"
#else
#include "grovekit.hpp"
#endif

#include "../lib/restclient-cpp/include/restclient-cpp/restclient.h"
#include "../lib/json-cpp/json/json.h"

const string DEFAULT_LATITUDE = "47.641944";
const string DEFAULT_LONGITUDE = "-122.127222";

// Convenience function to return the configured latitude
string latitude() {
  if (getenv("LATITUDE")) {
    return getenv("LATITUDE");
  } else {
    return DEFAULT_LATITUDE;
  }
}

// Convenience function to return the configured latitude
string longitude() {
  if (getenv("LONGITUDE")) {
    return getenv("LONGITUDE");
  } else {
    return DEFAULT_LONGITUDE;
  }
}

// Make a REST API call to the USGS to see if there has been
// a recent earthquake detected in the local area
void verify(Devices* devices) {
  devices->checking();

  // we'll check for quakes in the last ten minutes
  time_t base = time(NULL);
  struct tm* tm = localtime(&base);
  tm->tm_min -= 10;
  time_t next = mktime(tm);
  char mbstr[sizeof "2011-10-08T07:07:09Z"];
  strftime(mbstr, sizeof(mbstr), "%FT%TZ", localtime(&next));

  stringstream query;
  query << "http://earthquake.usgs.gov/fdsnws/event/1/query?";
  query << "format=geojson&";
  query << "starttime=" << mbstr << "&";
  query << "latitude=" << latitude() << "&";
  query << "longitude=" << longitude() << "&";
  query << "maxradiuskm=500";

  cerr << query.str() << endl;

  RestClient::response r = RestClient::get(query.str());
```

```
  Json::Value root;
  istringstream str(r.body);
  str >> root;

  const Json::Value features = root["features"];
  if (features.size() > 0) {
    devices->warning();
  } else {
    devices->noquake();
  }
}

Devices devices;

// Exit handler for program
void exit_handler(int param)
{
  devices.cleanup();
  exit(1);
}

// The main function for the example program
int main()
{
  // handles ctrl-c or other orderly exits
  signal(SIGINT, exit_handler);

  // create and initialize UPM devices
  devices.init();
  devices.reset();

  bool motionDetected = false;
  bool prev = false;

  for (;;) {
    motionDetected = devices.getAcceleration();

    if (motionDetected && !prev) { verify(&devices); }
    prev = motionDetected;
    sleep(1);
  }

  return MRAA_SUCCESS;
}

/**
 * @brief The function that would be called whenever a PUBLISH is received.
 *
 * @note This function is not used in this example.
 */
void publish_callback(void** unused, struct mqtt_response_publish *published);

/**
 * @brief The client's refresher. This function triggers back-end routines to
 *        handle ingress/egress traffic to the broker.
 *
 * @note All this function needs to do is call \ref __mqtt_recv and
 *       \ref __mqtt_send every so often. I've picked 100 ms meaning that
 *       client ingress/egress traffic will be handled every 100 ms.
 */
void* client_refresher(void* client);

/**
```

```
 * @brief Safelty closes the \p sockfd and cancels the \p client_daemon before \c
exit.
 */
void exit_example(int status, int sockfd, pthread_t *client_daemon);

/**
 * A simple program to that publishes the current time whenever ENTER is pressed.
 */
int main(int argc, const char *argv[])
{
    const char* addr;
    const char* port;
    const char* topic;

    /* get address (argv[1] if present) */
    if (argc > 1) {
        addr = argv[1];
    } else {
        addr = "test.mosquitto.org";
    }

    /* get port number (argv[2] if present) */
    if (argc > 2) {
        port = argv[2];
    } else {
        port = "1883";
    }

    /* get the topic name to publish */
    if (argc > 3) {
        topic = argv[3];
    } else {
        topic = "datetime";
    }

    /* open the non-blocking TCP socket (connecting to the broker) */
    int sockfd = open_nb_socket(addr, port);

    if (sockfd == -1) {
        perror("Failed to open socket: ");
        exit_example(EXIT_FAILURE, sockfd, NULL);
    }

    /* setup a client */
    struct mqtt_client client;
    uint8_t sendbuf[2048]; /* sendbuf should be large enough to hold multiple
whole mqtt messages */
    uint8_t recvbuf[1024]; /* recvbuf should be large enough any whole mqtt
message expected to be received */
    mqtt_init(&client, sockfd, sendbuf, sizeof(sendbuf), recvbuf,
sizeof(recvbuf), publish_callback);
    /* Create an anonymous session */
    const char* client_id = NULL;
    /* Ensure we have a clean session */
    uint8_t connect_flags = MQTT_CONNECT_CLEAN_SESSION;
    /* Send connection request to the broker. */
    mqtt_connect(&client, client_id, NULL, NULL, 0, NULL, NULL, connect_flags,
400);

    /* check that we don't have any errors */
    if (client.error != MQTT_OK) {
        fprintf(stderr, "error: %s\n", mqtt_error_str(client.error));
        exit_example(EXIT_FAILURE, sockfd, NULL);
    }
```

```
    /* start a thread to refresh the client (handle egress and ingree client
traffic) */
    pthread_t client_daemon;
    if(pthread_create(&client_daemon, NULL, client_refresher, &client)) {
        fprintf(stderr, "Failed to start client daemon.\n");
        exit_example(EXIT_FAILURE, sockfd, NULL);

    }

    /* start publishing the time */
    printf("%s is ready to begin publishing the time.\n", argv[0]);
    printf("Press ENTER to publish the current time.\n");
    printf("Press CTRL-D (or any other key) to exit.\n\n");
    while(fgetc(stdin) == '\n') {
        /* get the current time */
        time_t timer;
        time(&timer);
        struct tm* tm_info = localtime(&timer);
        char timebuf[26];
        strftime(timebuf, 26, "%Y-%m-%d %H:%M:%S", tm_info);

        /* print a message */
        char application_message[256];
        snprintf(application_message, sizeof(application_message), "The time is
%s", timebuf);
        printf("%s published : \"%s\"", argv[0], application_message);

        /* publish the time */
        mqtt_publish(&client, topic, application_message,
strlen(application_message) + 1, MQTT_PUBLISH_QOS_0);

        /* check for errors */
        if (client.error != MQTT_OK) {
            fprintf(stderr, "error: %s\n", mqtt_error_str(client.error));
            exit_example(EXIT_FAILURE, sockfd, &client_daemon);
        }
    }

    /* disconnect */
    printf("\n%s disconnecting from %s\n", argv[0], addr);
    sleep(1);

    /* exit */
    exit_example(EXIT_SUCCESS, sockfd, &client_daemon);
}

void exit_example(int status, int sockfd, pthread_t *client_daemon)
{
    if (sockfd != -1) close(sockfd);
    if (client_daemon != NULL) pthread_cancel(*client_daemon);
    exit(status);
}



void publish_callback(void** unused, struct mqtt_response_publish *published)
{
    /* not used in this example */
}
```

Додаток В

СТВОРЕННЯ ІОТ СИСТЕМИ МОНІТОРИНГУ СЕЙСМОЛОГІЧНОЇ СИТУАЦІЇ НА ПРИКЛАДІ ОДНОГО З РЕГІОНІВ УКРАЇНИ

Текст програми для ОС Windows

Листів 23

Розробник _____ Дмитро САХАРОВ

Н

Київ, 2021

```
from tkinter import *
import pymysql as mdb
import pandas as pd
import sys
import os
import re
import datetime
import time
from time import sleep
import numpy as np
import collections
import matplotlib
matplotlib.use('TkAgg')
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle



class PlotInterface(object):
    def __init__(self, df, EvT_color_set, EvT_size_set):
        self.df = df
        self.ax = plt.gca()
        self.rect = Rectangle((0.1,0.1), 0, 0, fill=False)
        self.rect.set_visible(False)
        self.ax.add_patch(self.rect)
        self.x0 = None
        self.y0 = None
        self.x1 = None
        self.y1 = None
        self.press = False
```

```
    self.sub_df = None
    self.EvT_color_set = EvT_color_set
    self.EvT_size_set = EvT_size_set
    self.ax.figure.canvas.mpl_connect('button_press_event', self.on_press)
    self.ax.figure.canvas.mpl_connect('motion_notify_event',self.on_motion)
    self.ax.figure.canvas.mpl_connect('button_release_event', self.on_release)

def on_press(self, event):
    if event.inaxes:

        self.press = True
        self.x0 = event.xdata
        self.y0 = event.ydata
        self.rect.set_width(0)
        self.rect.set_height(0)
        self.rect.set_xy((self.x0,self.y0))
        self.rect.set_visible(True)
        #Save the plot background
        self.background = self.ax.figure.canvas.copy_from_bbox(self.ax.bbox)



def on_release(self, event):
    if event.inaxes:
        self.press = False
        self.x1 = event.xdata
        self.y1 = event.ydata
        self.rect.set_visible(False)
        self.ax.figure.canvas.draw()
        self.min_x = min(self.x0,self.x1)
```

```
            self.max_x = max(self.x0,self.x1)
            self.min_y = min(self.y0,self.y1)
            self.max_y = max(self.y0,self.y1)

            if self.sub_df is not None:
                self.sub_df = self.sub_df.append(self.df[ (self.df['timestamp'] > self.min_x) & (self.df['timestamp'] < self.max_
& \
                        (self.df['amplitude'] > self.min_y) & (self.df['amplitude'] < self.max_y)])
            else:
                self.sub_df = self.df[ (self.df['timestamp'] > self.min_x) & (self.df['timestamp'] < self.max_x) & \
                        (self.df['amplitude'] > self.min_y) & (self.df['amplitude'] < self.max_y)]

            plt.scatter(self.sub_df['timestamp'],self.sub_df['amplitude'],color='blue',s=self.EvT_size_set)
            self.ax.figure.canvas.draw()

            print('Selected (%d < timestamp < %d) and (%d < amplitude < %d)' % (self.min_x, self.max_x, self.min_
self.max_y))

    def on_motion(self, event):
        if self.press is False: return
        if event.inaxes != self.rect.axes: return
        self.x1 = event.xdata
        self.y1 = event.ydata
        self.rect.set_width(self.x1 - self.x0)
        self.rect.set_height(self.y1 - self.y0)
        self.rect.set_xy((self.x0, self.y0))
        self.ax.figure.canvas.restore_region(self.background)
        self.ax.draw_artist(self.rect)
        self.ax.figure.canvas.blit(self.rect.clipbox)
```

```
class App(object):
    def __init__(self, window, ip, user, pswd, db):
        self.window = window
        self.ip = ip
        self.user = user
        self.pswd = pswd
        self.db = db

        window.wm_title("Radon Database Interface")

        self.current_row=0

        '''Query Panel'''
        #Run tag field
        self.runtag_label = Label (window, text= "Data Run Tag: ")
        self.runtag_label.grid(row=self.current_row,column=0)
        self.runtag_text = StringVar()
        Entry(window, textvariable=self.runtag_text).grid(row=self.current_row,column=1)
        self.current_row += 1

        #Start time field
        self.start_t_label = Label (window, text= "Start Time (YYYY-MM-DD HH:MM:SS): ")
        self.start_t_label.grid(row=self.current_row,column=0)
        self.start_t_text = StringVar()
        Entry(window, textvariable=self.start_t_text).grid(row=self.current_row,column=1)
        self.current_row += 1

        #End time field
        self.end_t_label = Label (window, text= "End Time (YYYY-MM-DD HH:MM:SS): ")
        self.end_t_label.grid(row=self.current_row, column=0)
```

```
        self.end_t_text = StringVar()
        Entry(window, textvariable=self.end_t_text).grid(row=self.current_row,column=1)
        self.current_row += 1

        #Limit field
        self.limit_label = Label (window, text= "Query Limit (Integer)")
        self.limit_label.grid(row=self.current_row, column=0)
        self.limit_text = StringVar()
        Entry(window, textvariable=self.limit_text).grid(row=self.current_row,column=1)
        self.current_row += 1

        #Send Query button
        self.query_button = StringVar()
        self.query_button.set("Send Query")
        Button(window, textvariable=self.query_butto
command=self.sendQuery).grid(row=self.current_row,column=0,columnspan=2)
        self.current_row += 1

        #Query status
        self.querystatus = Label(window, text='')
        self.querystatus.grid(row=self.current_row,column=0,columnspan=2)
        self.current_row += 1

        '''Line to separate segments'''
        canvas = Canvas(master=window, width=500, height=40)
        canvas.create_line(0, 20, 500, 20, fill="black")
        canvas.grid(row=self.current_row,column=0,columnspan=2)
        self.current_row += 1

        '''Plot Panel'''
```

```
#Drop down menu for time units
Label(window, text="Time Units").grid(row = self.current_row, column = 0, columnspan=2)
self.current_row += 1

self.timevar = StringVar(window)
# Use dictionary to map time unit to second -> unit conversion factor
self.choices = collections.OrderedDict()
self.choices['Seconds'] = 1
self.choices['Minutes'] = 1/60
self.choices['Hours'] = 1/3600
self.choices['Days'] = 1/86400
self.choices['Weeks'] = 1/604800

self.timevar.set('Seconds')
self.unit = self.timevar.get()
self.unit_conv = self.choices[self.unit]
self.option = OptionMenu(window, self.timevar, *self.choices, command = self.updateTimeUnit)

self.option.grid(row = self.current_row, column=0, columnspan=2)
#self.unit_conv = self.choices[self.timevar.get()]
self.current_row += 1



#EvT Plot
self.EvT_color = Label (window, text= "Data Color:")
self.EvT_color.grid(row=self.current_row,column=0)
self.EvT_size = Label (window, text= "Data Size:")
self.EvT_size.grid(row=self.current_row,column=1)
self.current_row += 1
```

```
self.EvT_color_text = StringVar(value='black')
EvT_Entry = Entry(window,textvariable=self.EvT_color_text).grid(row=self.current_row,column=0)
self.EvT_size_text = IntVar(value=10)
Entry(window, textvariable=self.EvT_size_text).grid(row=self.current_row,column=1)
self.current_row += 1

self.EvT_button = Button (window, text="Energy v Time Plot", command=self.EvT_plot)
self.EvT_button.grid(row=self.current_row,column=0,columnspan=2)
self.current_row += 1



#Energy Histogram Plot
self.EHist_color = Label (window, text= "Hist Color:")
self.EHist_color.grid(row=self.current_row,column=0)
self.EHist_bin = Label (window, text= "Hist Binning:")
self.EHist_bin.grid(row=self.current_row,column=1)
self.current_row += 1

self.EHist_color_text = StringVar(value='black')
Entry(window, textvariable=self.EHist_color_text).grid(row=self.current_row,column=0)
self.EHist_bin_text = IntVar(value=20)
Entry(window, textvariable=self.EHist_bin_text).grid(row=self.current_row,column=1)
self.current_row += 1

self.EHist_button = Button (window, text="Energy Histogram", command=self.EHist_plot)
self.EHist_button.grid(row=self.current_row,column=0,columnspan=2)
self.current_row += 1
```

```
#Time Histogram Settings (replace with error bar of count v time)
self.THist_color = Label (window, text= "Hist Color:")
self.THist_color.grid(row=self.current_row,column=0)
self.THist_bin = Label (window, text= "Hist Binning:")
self.THist_bin.grid(row=self.current_row,column=1)
self.current_row += 1

self.THist_color_text = StringVar(value='black')
Entry(window, textvariable=self.THist_color_text).grid(row=self.current_row,column=0)
self.THist_bin_text = IntVar(value=20)
Entry(window, textvariable=self.THist_bin_text).grid(row=self.current_row,column=1)
self.current_row += 1

self.THist_button = Button (window, text="Time Histogram", command=self.THist_plot)
self.THist_button.grid(row=self.current_row,column=0,columnspan=2)
self.current_row += 1

'''Line to separate segments'''
canvas = Canvas(master=window, width=500, height=40)
canvas.create_line(0, 20, 500, 20, fill="black")
canvas.grid(row=self.current_row,column=0,columnspan=2)
self.current_row += 1


'''Event Selection Panel'''
self.select_button = Button (window, text="Select Data from EvT", command=self.EvT_select)
self.select_button.grid(row=self.current_row,column=0,columnspan=1)

self.cancel_button = Button (window, text="Cancel Selection", command=self.EvT_select_cancel)
self.cancel_button.grid(row=self.current_row,column=1,columnspan=1)
```

```
self.current_row += 1

self.delete_button = Button (window, text="Delete Selection", command=self.EvT_delete)
self.delete_button.grid(row=self.current_row,column=0,columnspan=1)

self.select_only_button = Button (window, text="Delete All But Selection", command=self.EvT_select_only)
self.select_only_button.grid(row=self.current_row,column=1,columnspan=1)
self.current_row += 1


'''Line to separate segments'''
canvas = Canvas(master=window, width=500, height=40)
canvas.create_line(0, 20, 500, 20, fill="black")
canvas.grid(row=self.current_row,column=0,columnspan=2)
self.current_row += 1


'''Saving Panel'''
self.save_label = Label (window, text= "Save Location: ")
self.save_label.grid(row=self.current_row, column=0, columnspan=2)
self.current_row += 1

self.save_text = StringVar()
Entry(window, textvariable=self.save_text).grid(row=self.current_row,column=0, columnspan=2)
self.current_row += 1

self.save_button = Button (window, text="Save Data", command=self.save_to_csv)
self.save_button.grid(row=self.current_row,column=0,columnspan=2)
self.current_row += 1
```

```
'''Plot Setting Functions'''
#Change timestamp units
def updateTimeUnit(self,event):
    #Verify that the data frame is not empty
    if self.df.empty:
        messagebox.showinfo('Selection Error', 'Error: Please collect data before selecting time units.')
    else:
        self.unit = self.timevar.get()
        self.unit_conv = self.choices[self.unit]

        #Change units on timestamp plots if they exist
        self.df['timestamp']=self.df['orig_timestamp'] * self.unit_conv

        #Figure two is the EvT plot:
        if plt.fignum_exists(2):
            self.EvT_plot()

        #Figure one is the Time Histogram
        if plt.fignum_exists(1):
            self.THist_plot()

#Set Energy histogram options
def EHist_set (self):
    self.EHist_color_set = self.EHist_color_text.get()
    self.EHist_bin_set = self.EHist_bin_text.get()

#Set Time histogram options
def THist_set (self):
    self.THist_color_set = self.THist_color_text.get()
```

```
    self.THist_bin_set = self.THist_bin_text.get()

#Set EvT plotting options
def EvT_set (self):
    self.EvT_color_set = self.EvT_color_text.get()
    self.EvT_size_set = self.EvT_size_text.get()

'"Plotting Functions"'
#Energy Histogram
def EHist_plot (self):
    #Verify that the data frame is not empty
    if self.df.empty:
        messagebox.showinfo('Plot Error', 'Error: No data has been collected.  Can not produce empty plot.')
    else:
        self.EHist_color_set = self.EHist_color_text.get()
        self.EHist_bin_set = self.EHist_bin_text.get()

        plt.figure(0)
        plt.clf()
        plt.hist(self.df['amplitude'],self.EHist_bin_set, normed=0, facecolor=self.EHist_color_set)
        plt.title("Energy Histogram", fontsize=16)
        plt.ylabel("Count",fontsize=14)
        plt.xlabel("Energy",fontsize=14)
        plt.show(block=False)

#Time Histogram
def THist_plot (self):
    #Verify that the data frame is not empty
    if self.df.empty:
        messagebox.showinfo('Plot Error', 'Error: No data has been collected.  Can not produce empty plot.')
```

```
    else:
        self.THist_color_set = self.THist_color_text.get()
        self.THist_bin_set = self.THist_bin_text.get()

        #Energy V Time plot
        plt.figure(1)
        plt.clf()
        plt.hist(self.df['timestamp'],self.THist_bin_set, normed=0, facecolor=self.THist_color_set)
        plt.title("Time Histogram", fontsize=16)
        plt.ylabel("Count",fontsize=14)
        plt.xlabel(self.unit,fontsize=14)
        plt.show(block=False)

#Energy V Time plot
def EvT_plot (self):
    #Verify that the data frame is not empty
    if self.df.empty:
        messagebox.showinfo('Plot Error', 'Error: No data has been collected.  Can not produce empty plot.')
    else:
        self.EvT_color_set = self.EvT_color_text.get()
        self.EvT_size_set = self.EvT_size_text.get()

        #Energy V Time plot
        plt.figure(2)
        plt.clf()
        plt.scatter(self.df['timestamp'],self.df['amplitude'],color=self.EvT_color_set,s=self.EvT_size_set)
        plt.title("Energy v Time", fontsize=16)
        plt.ylabel("Amplitude",fontsize=14)
        plt.xlabel(self.unit,fontsize=14)
        plt.show(block=False)
```

```
'''Event Selection Functions'''
def EvT_select (self):
    #Verify that the data frame is not empty
    if self.df.empty:
        messagebox.showinfo('Plot Error', 'Error: No data has been collected.  Can not select absent data.')
    elif not plt.fignum_exists(2):
        messagebox.showinfo('Plot Error', 'Please open the EvT plot before trying to select data.')
    else:
        plt.figure(2)
        plt.scatter(self.df['timestamp'],self.df['amplitude'],color=self.EvT_color_set,s=self.EvT_size_set)
        self.EvT_interface = PlotInterface(self.df,self.EvT_color_set,self.EvT_size_set)

#Cancel event selection
def EvT_select_cancel (self):
    if self.EvT_interface:
        del self.EvT_interface
        self.EvT_plot()

def EvT_delete (self):
    #Verify that the data frame is not empty
    if self.df.empty:
        messagebox.showinfo('Plot Error', 'Error: No data has been collected.  Can not delete absent data.')
    else:
        #If we have selected some data, delete that data
        if self.EvT_interface:
            sub = ['timestamp', 'amplitude']
            mask = self.df[sub].isin(self.EvT_interface.sub_df[sub].to_dict(orient='list')).all(axis=1)
            self.df = self.df[~mask]
```

```
            self.EvT_interface = None

        #Update the plots
        #Figure two is the EvT plot:
        if plt.fignum_exists(2):
            self.EvT_plot()

        #Figure one is the Time Histogram
        if plt.fignum_exists(1):
            self.THist_plot()

        #Figure zero is the Energy Histogram
        if plt.fignum_exists(0):
            self.EHist_plot()

def EvT_select_only (self):
    #Verify that the data frame is not empty
    if self.df.empty:
        messagebox.showinfo('Plot Error', 'Error: No data has been collected.  Can not delete absent data.')
    else:
        #If we have selected some data, delete that data
        if self.EvT_interface:
            sub = ['timestamp', 'amplitude']
            mask = self.df[sub].isin(self.EvT_interface.sub_df[sub].to_dict(orient='list')).all(axis=1)
            self.df = self.df[mask]
            self.EvT_interface = None

        #Update the plots
        #Figure two is the EvT plot:
        if plt.fignum_exists(2):
```

```
            self.EvT_plot()

        #Figure one is the Time Histogram
        if plt.fignum_exists(1):
            self.THist_plot()

        #Figure zero is the Energy Histogram
        if plt.fignum_exists(0):
            self.EHist_plot()



'"Saving to CSV"'

#Could add more options for saving if desired... such as save to txt, or changing delimiter
def save_to_csv(self):
    #Verify that the data frame is not empty
    if self.df.empty:
        messagebox.showinfo('Save Error', 'Error: Can not save empty data frame')

    #If we have data
    else:
        #Get the save location and extract the directory path
        self.save_loc = self.save_text.get()
        self.save_dir = os.sep.join(self.save_loc.split(os.sep)[:-1])

        #Verify that the directory exists
        if os.path.isdir(self.save_dir):

            #Verify that the filename is more than just '.csv'
            if len(self.save_loc.split(os.sep)[-1]) < 5:
```

```
                messagebox.showinfo('Save Error', 'Error: Path should end in filename.csv')

            #Verify that the filename ends in '.csv'
            elif self.save_loc.split(os.sep)[-1][-4:] != '.csv':
                messagebox.showinfo('Save Error', 'Error: File should end in filename.csv')

            #Save the data if it passes error checking
            else:
                self.df.to_csv(self.save_loc)

        #Error if the directory doesn't exist
        else:
            messagebox.showinfo('Save Error', 'Error: File directory does not exist')

"'Query Functions'"
def sendQuery(self):
    start_t = self.start_t_text.get()
    end_t = self.end_t_text.get()
    run_tag = self.runtag_text.get()
    limit = self.limit_text.get()

    #Check that date is valid
    time_format = re.compile('^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}$')
    start_t_okay=None
    if (start_t) and (time_format.match(start_t)):
        start_year=int(start_t[0:4])
        start_month=int(start_t[5:7])
        start_day=int(start_t[8:10])
        start_hour=int(start_t[11:13])
        start_min=int(start_t[14:16])
```

```
            start_sec=int(start_t[17:19])
            try:
                datetime.datetime(start_year,start_month,start_day,start_hour,start_min,start_sec)
                start_t_okay='yes'
            except:
                start_t_okay='no'



        end_t_okay=None
        if (end_t) and (time_format.match(end_t)):
            end_year=int(end_t[0:4])
            end_month=int(end_t[5:7])
            end_day=int(end_t[8:10])
            end_hour=int(end_t[11:13])
            end_min=int(end_t[14:16])
            end_sec=int(end_t[17:19])
            try:
                datetime.datetime(end_year,end_month,end_day,end_hour,end_min,end_sec)
                end_t_okay='yes'
            except:
                end_t_okay='no'


        '''Error Handling'''
        #Check that the date is in the correct format
        if (start_t) and (not time_format.match(start_t)):
            messagebox.showinfo('Query Error', 'Error: Start time Must have the format "YYYY-MM-DD HH:MM:SS" or l
empty')
        elif (end_t) and (not time_format.match(end_t)):
            messagebox.showinfo('Query Error', 'Error: End time Must have the format "YYYY-MM-DD HH:MM:SS" or l
empty')
        #Check that the date is a valid date
        elif (start_t) and (start_t_okay == 'no'):
```

```
    messagebox.showinfo('Query Error', 'Error: Start time is not valid')
elif (end_t) and (end_t_okay == 'no'):
    messagebox.showinfo('Query Error', 'Error: End time is not valid')
#Check that start_t < end_t
elif (start_t) and (end_t) and (start_t > end_t):
    messagebox.showinfo('Query Error', 'Start time must come before end time')
else:
    #Default to not including the conditions in the query
    run_tag_string = ''
    time_string = ''
    limit_string = ''

    #Toggle query conditions on if the GUI field isn't blank

    #Run tag condition
    if run_tag != '':
        run_tag_string = 'AND run_tag = "%s"' %(run_tag)

    #Time window condition (in order: both set, only start set, only end set)
    if start_t != '' and end_t != '':
        time_string = 'AND clock_time BETWEEN "%s" AND "%s"' %(start_t,end_t)
    elif start_t != '':
        time_string = 'AND clock_time > "%s"' %(start_t)
    elif end_t != '':
        time_string = 'AND clock_time < "%s"' %(end_t)

    #Limit query
    if limit != '':
        limit_string = 'LIMIT %s' %(limit)
```

```
        #Setting the query
        #We don't really need "amplitude IS NOT NULL" condition, it was added so that
        #we can add conditions with simple 'and ...'
        #This way we don't have to worry about whether or not to include a 'where' clause
        self.query = "SELECT clock_time, amplitude FROM summarydata where amplitude IS NOT NULL %s %s %s" (run_tag_string, time_string, limit_string)

        con = mdb.connect(self.ip, self.user, self.pswd, self.db);

        #With will close the connection after the code is done,
        #regardless of how the code exists. Use as an alternative to 'finally' statement
        with con:

            cur = con.cursor()

            #Send select statement
            #columns are [id, 'run_tag', clock_time, centroid_time, amplitude, rmse]
            cur.execute(self.query)

            #Get all of the MySQL return at once
            #Returns a tuple of tuples, with each inner tupple being one row
            self.df = pd.DataFrame()
            rows = cur.fetchall()
            self.df['timestamp']=[time.mktime(rows[i][0].timetuple()) for i in range(len(rows))]

            #Set up default 1 second time stamps
            self.start_timestamp =  min(self.df['timestamp'])
            self.df['timestamp'] = self.df['timestamp'] - self.start_timestamp
            self.df['orig_timestamp']=self.df['timestamp']
            self.df['amplitude'] = [rows[i][1] for i in range(len(rows))]
```

```
        self.querystatus.configure(text = "Finished")

        #Check if the query returned data
        if self.df.empty:
            messagebox.showinfo('Query Warning', 'Warning: Query returned no data')


#Use a main wrapper so that the code isn't executed if it is ever imported as a module
#Arguements need to be IP of database,
def main(argv):
'''
Usage: python radonGUI.py databaseIP username password databaseTable
'''
ip = argv[1]
user = argv[2]
pswd = argv[3]
db = argv[4]

window= Tk()
start= App(window,ip,user,pswd,db)
window.mainloop()

if __name__ == "__main__":
   main(sys.argv)
```

Файл gui.py

```
Add-Type -AssemblyName System.Windows.Forms
[System.Windows.Forms.Application]::EnableVisualStyles()

$Form                    = New-Object system.Windows.Forms.Form
$Form.ClientSize         = '400,491'
$Form.text               = "EarthShake"
$Form.TopMost            = $false

$Button1                 = New-Object system.Windows.Forms.Button
$Button1.text            = "Отримати"
$Button1.width           = 72
$Button1.height          = 30
$Button1.location        = New-Object System.Drawing.Point(159,441)
$Button1.Font            = 'Microsoft Sans Serif,10'

$TextBox1                = New-Object system.Windows.Forms.TextBox
$TextBox1.multiline      = $false
$TextBox1.width          = 100
$TextBox1.height         = 20
$TextBox1.location       = New-Object System.Drawing.Point(274,326)
$TextBox1.Font           = 'Microsoft Sans Serif,10'

$Historicaldata          = New-Object system.Windows.Forms.Label
$Historicaldata.text     = "Історичні дані"
```

```
$Historicaldata.AutoSize        = $true
$Historicaldata.width           = 25
$Historicaldata.height          = 10
$Historicaldata.location        = New-Object System.Drawing.Point(259,7)
$Historicaldata.Font            = 'Microsoft Sans Serif,10,style=Underline'

$Label1                     = New-Object system.Windows.Forms.Label
$Label1.text                = "Дані в реальному часі"
$Label1.AutoSize            = $true
$Label1.width               = 25
$Label1.height              = 10
$Label1.location            = New-Object System.Drawing.Point(30,7)
$Label1.Font                = 'Microsoft Sans Serif,10'

$DataGridView1              = New-Object system.Windows.Forms.DataGridView
$DataGridView1.width        = 338
$DataGridView1.height       = 250
$DataGridView1.location     = New-Object System.Drawing.Point(36,32)

$ToolTip1                   = New-Object system.Windows.Forms.ToolTip

$TextBox2                   = New-Object system.Windows.Forms.TextBox
$TextBox2.multiline         = $false
$TextBox2.width             = 100
$TextBox2.height            = 20
$TextBox2.location          = New-Object System.Drawing.Point(36,326)
```

```
$TextBox2.Font                  = 'Microsoft Sans Serif,10'

$Label2                     = New-Object system.Windows.Forms.Label
$Label2.text                 = "Дата, з:"
$Label2.AutoSize               = $true
$Label2.width                 = 25
$Label2.height                = 10
$Label2.location               = New-Object System.Drawing.Point(38,301)
$Label2.Font                  = 'Microsoft Sans Serif,10'

$Label3                     = New-Object system.Windows.Forms.Label
$Label3.text                 = "Дата, по"
$Label3.AutoSize               = $true
$Label3.width                 = 25
$Label3.height                = 10
$Label3.location               = New-Object System.Drawing.Point(282,301)
$Label3.Font                  = 'Microsoft Sans Serif,10'

$RadioButton1                 = New-Object system.Windows.Forms.RadioButton
$RadioButton1.text             = "кількість землетрусів відповідної інтенсивності "
$RadioButton1.AutoSize           = $true
$RadioButton1.width             = 104
$RadioButton1.height            = 20
$RadioButton1.location           = New-Object System.Drawing.Point(36,373)
$RadioButton1.Font              = 'Microsoft Sans Serif,10'
```

```
$RadioButton2                = New-Object system.Windows.Forms.RadioButton

$RadioButton2.text           = "кількість землетрусів  по днях"

$RadioButton2.AutoSize       = $true

$RadioButton2.width          = 104

$RadioButton2.height         = 20

$RadioButton2.location       = New-Object System.Drawing.Point(36,404)

$RadioButton2.Font           = 'Microsoft Sans Serif,10'


$Form.controls.AddRange(@($Button1,$TextBox1,$Historicaldata,$Label1,$DataGridView1,$TextBox2,$Label2,$Label3,$RadioButton1,$RadioButton2))

$Historicaldata.Add_Click({  })
```

Додаток Г

СТВОРЕННЯ ІОТ СИСТЕМИ МОНІТОРИНГУ СЕЙСМОЛОГІЧНОЇ СИТУАЦІЇ НА ПРИКЛАДІ ОДНОГО З РЕГІОНІВ УКРАЇНИ

Перелік елементів

Листів 2

Розробник _____ Дмитро САХАРОВ

Н

Київ, 2021

Поз. позначення	Найменування	Кіл.	Примітка
	Конденсатори		
C1	C-0805 10 В 10 мкФ 5% X5R Faithful Link	1	
C2,C3	C-0805 50 В 20 пФ 5% X5R Faithful Link	2	
C4,C5	C-0805 50 В 100 нФ 5% X5R Faithful Link	2	
C6	C-0805 10 В 10 мкФ 5% X5R Faithful Link	1	
C7	C-0805 50 В 10 мкФ 5% X5R Faithful Link	1	
C8–C11	C-0805 50 В 10 нФ 5% X5R Faithful Link	4	
C12	C-0805 10 В 10 мкФ 5% X5R Faithful Link	1	
C13	C-0805 50 В 0,1 мкФ 5% X5R Faithful Link	1	
C14,C15	C-0805 10 В 10 мкФ 5% X5R Faithful Link	2	
	Мікросхеми		
DD1	TP4056 NanJing Top Power	1	
DD2	STM32F401VDH6 STMicroelectronics	1	
DD3	TPS63001 Texas Instruments	1	
DD4	MT3608 Aerosemi Technology	1	
	Пристрої індикації		
HL1	LTST-C191KRKT Lite-On	1	
HL2	LTST-C190GKT Lite-On	1	

Перв. застос. IPм11.402221.001

Справ. № IPм11.402221.001

Підп. та дата

Інв. № дубл.

Взам. інв. №

Підп. та дата

Інв. № подл.

Зм.	Лист	№ докум.	Підп.	Дата
Розробив		Сахаров Д.Ю.		
Перевірив		Кучеренко Р.Ю.		
Н.контр		Кравченко О.В.		
Затв.		Кравченко О.В.		

IPм11.402221.001 ПЕ3

Сейсмограф

Перелік елементів

Літ.	Аркуш	Аркушів
	1	2

КНУ ім. Тараса Шевченка
ФІТ, IPм-11

Поз. позна-чення	Найменування	Кіл.	Примітка
	Катушки індуктивності		
L1	L-1210 2,2 мкГн 570 мА B82422H1682K000 Epcos	1	
L2	L-1210 22 мкГн 570 мА B82422H1682K000 Epcos	1	
	Резистори		
R1,R2	R-0805 0,125 4,7 кОм 5% Yageo	2	
R3	R-0805 0,125 1.2 кОм 5% Yageo	1	
R4	R-0805 0,125 0,4 Ом 5% Yageo	1	
R5	R-0805 0,125 10 кОм 5% Yageo	1	
R6	R-0805 0,125 10 кОм 5% Yageo	1	
	Кнопки		
SB1	SWT-6/4,3 KLS	1	
	Роз'єми		
XS1	5025850270 Molex	1	
XS2	ZX62-AB-5PA Hirose Electric	1	
XS3	PLS 6x1 CUI Inc	1	
	Генератори		
ZQ1	TXC CORPORATION	1	

IPм11402221001

Підп. та дата | Інв. № дубл. | Взам. інв.№ | Підп. та дата | Інв. № підл.

Зм.	Арк.	№ докум.	Підп.	Дата	IPм11.402221.001 ПЕЗ	Арк. 2

Додаток Д

СТВОРЕННЯ ІОТ СИСТЕМИ МОНІТОРИНГУ СЕЙСМОЛОГІЧНОЇ СИТУАЦІЇ НА ПРИКЛАДІ ОДНОГО З РЕГІОНІВ УКРАЇНИ

Графічні зображення

Листів 6

Розробник _____ Дмитро САХАРОВ

Н

Київ, 2021

Зміст

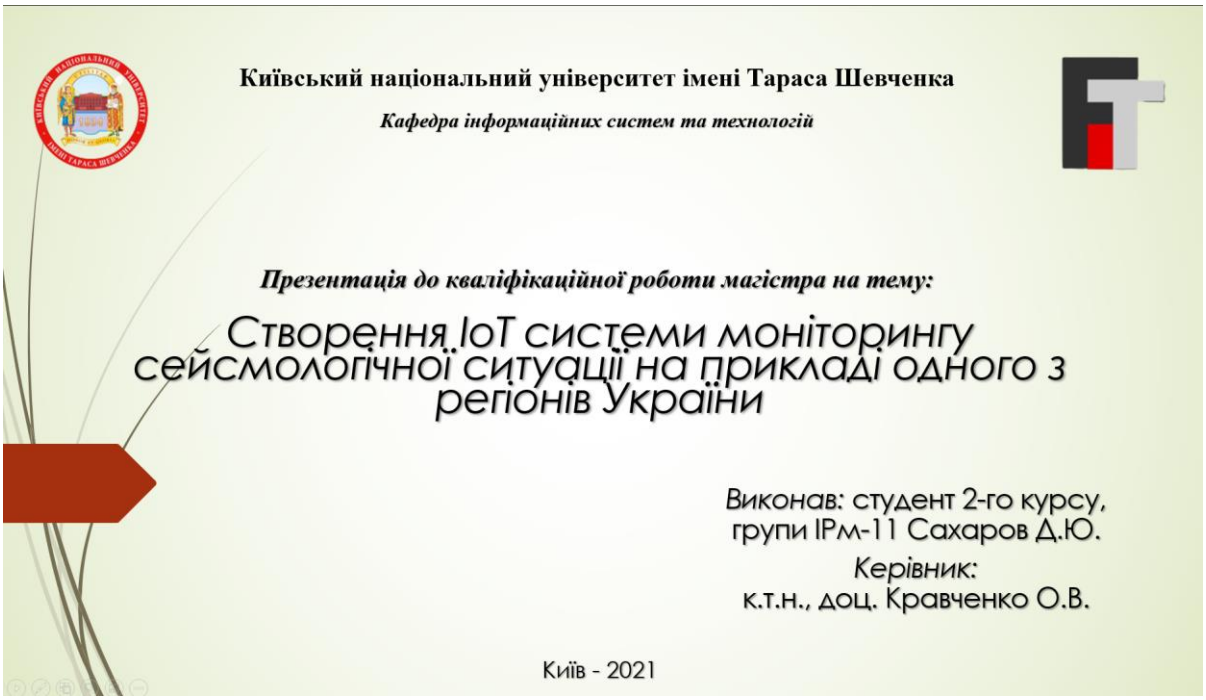

Рисунок В.1 – Слайд 1. Титульна сторінка

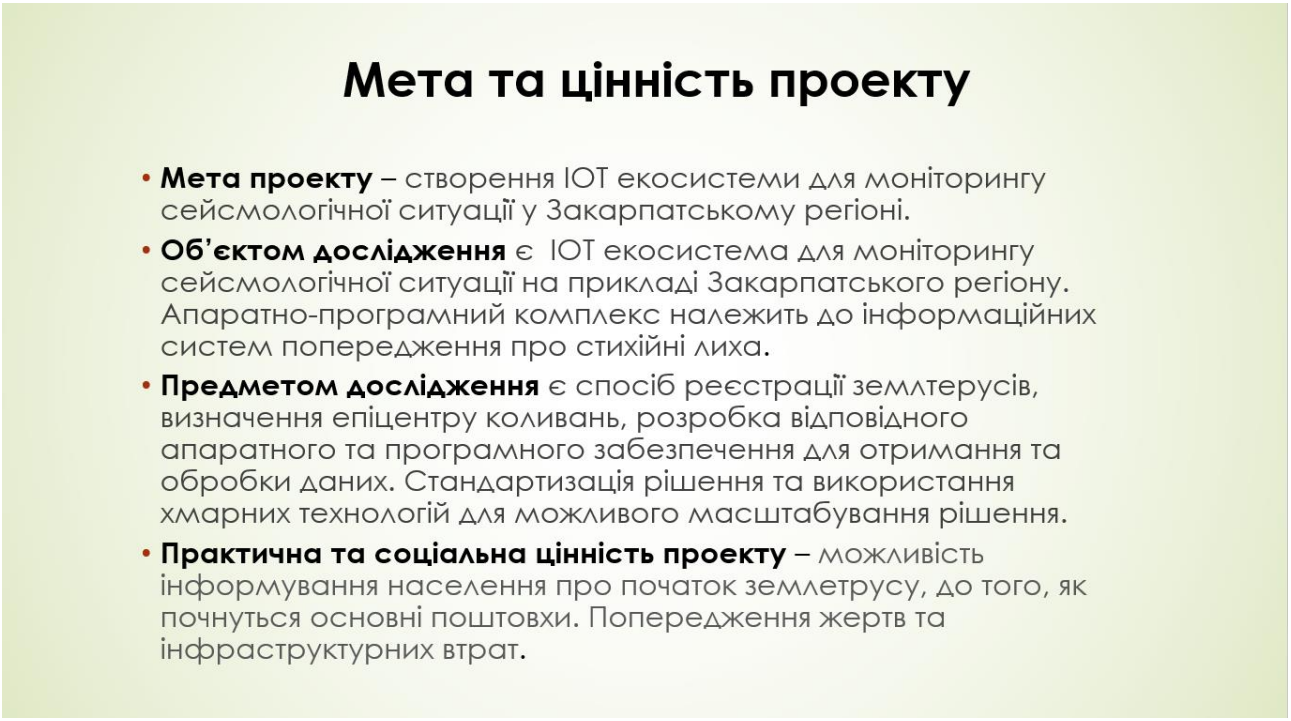

Рисунок В.2 – Слайд 2. Мета та цінність

Для досягнення мети необхідно

- провести аналіз існуючих систем інформування про стихійні лиха;
- виконати дослідження технології зв'язку для даної системи;
- сформувати уявлення про конструктивні та архітектурні рішення;
- дослідити алгоритми детектування землетрусів, та локалізації епіцентрів;
- інтегрувати хмарні сервіси у систему;
- описати логічні зв'язки вузлів системи на етапі проектування;
- розробити схему принципову IoT сейсмографа;
- виконати верифікацію даних;
- підготувати публікацію до тематичної конференції.

Рисунок В.3 – Слайд 3. План досягнення мети

Огляд аналогів

Параметри	Аналог 1	Аналог 2	Аналог 3
Назва	Earthquake Early Warning System by IOT using Wireless Sensor Networks	A Smart IoT Device for Detecting and Responding to Earthquakes	Technologies of Internet of Things applied to an Earthquake Early Warning System
Країна	Індія	Південна Корея	Іспанія
Переваги	Добре описано процес тестування та валідації приладу сейсмографу.	Алгоритм детектування, що базується на нейромережі. Хороші приклади побудови IoT екосистеми на прикладі домашньої автоматики	Рішення виконано як додаток для смартфону, та використовує наявні ресурси смартфону для детектування сейсмоактивності.
Недоліки	Невдалий вибір протоколу зв'язку.	Рішення підходить тільки для домашньої автоматики – оскільки для передачі даних використовується Wi-Fi.	Низька густина покриття сенсорів у малонаселених районах. Високий показник шуму через коритуваць ку активність.

Рисунок В.4 – Слайд 4. Огляд аналогів

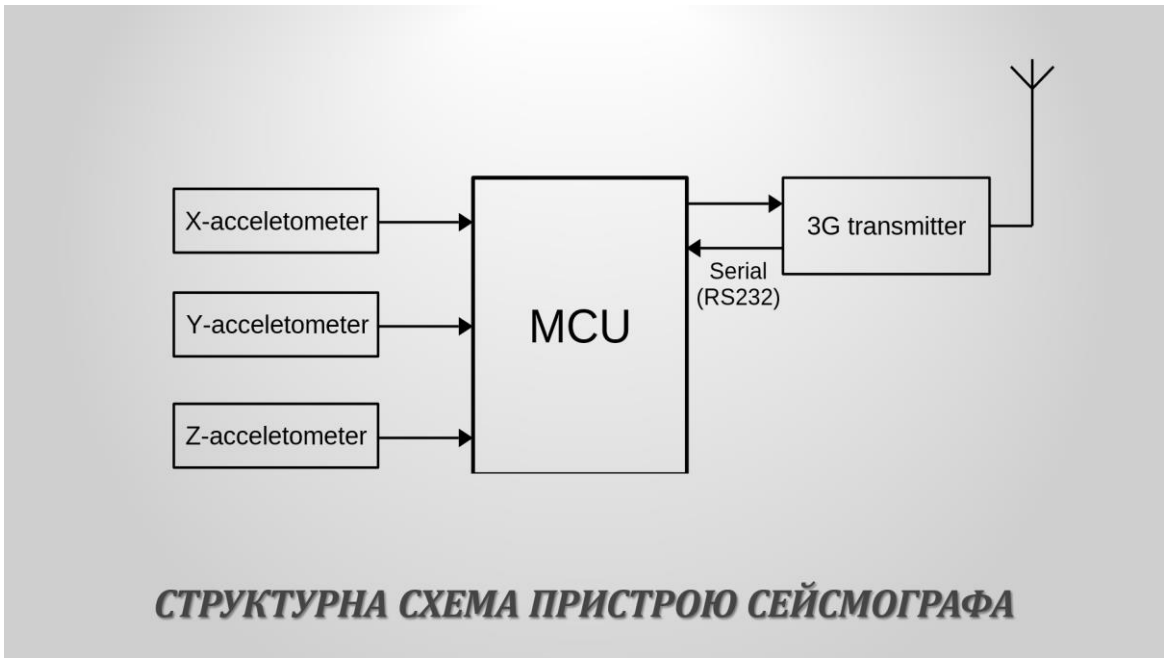

Рисунок В.5 – Слайд 5. Структурна схема

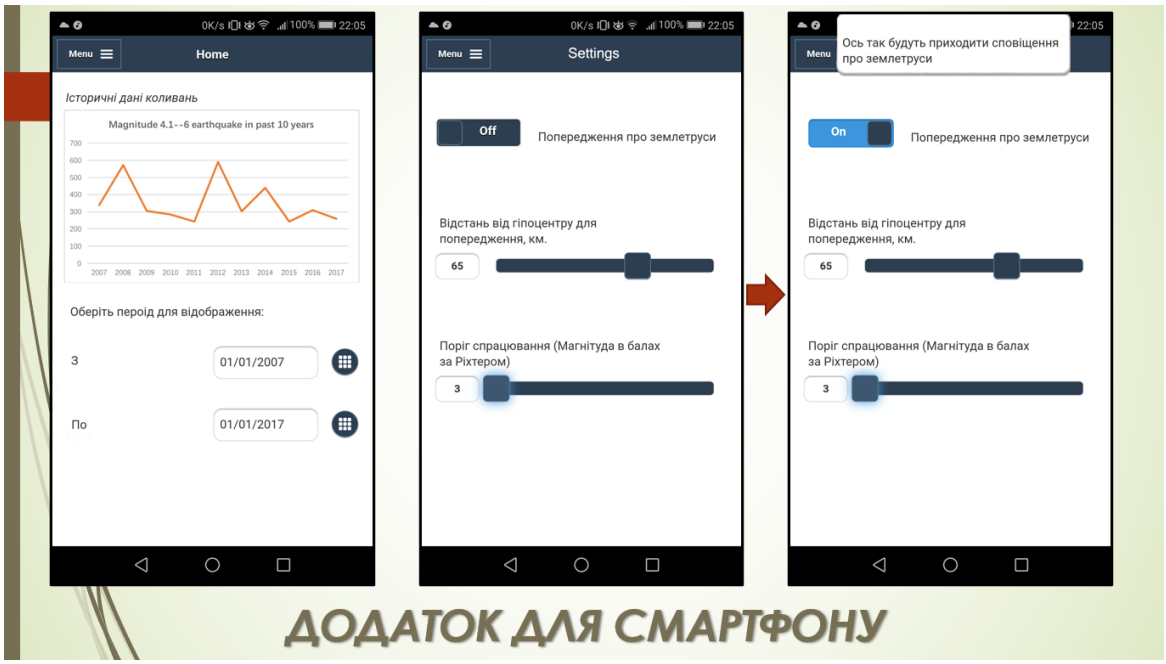

Рисунок В.6 – Слайд 6. Додаток для смартфону

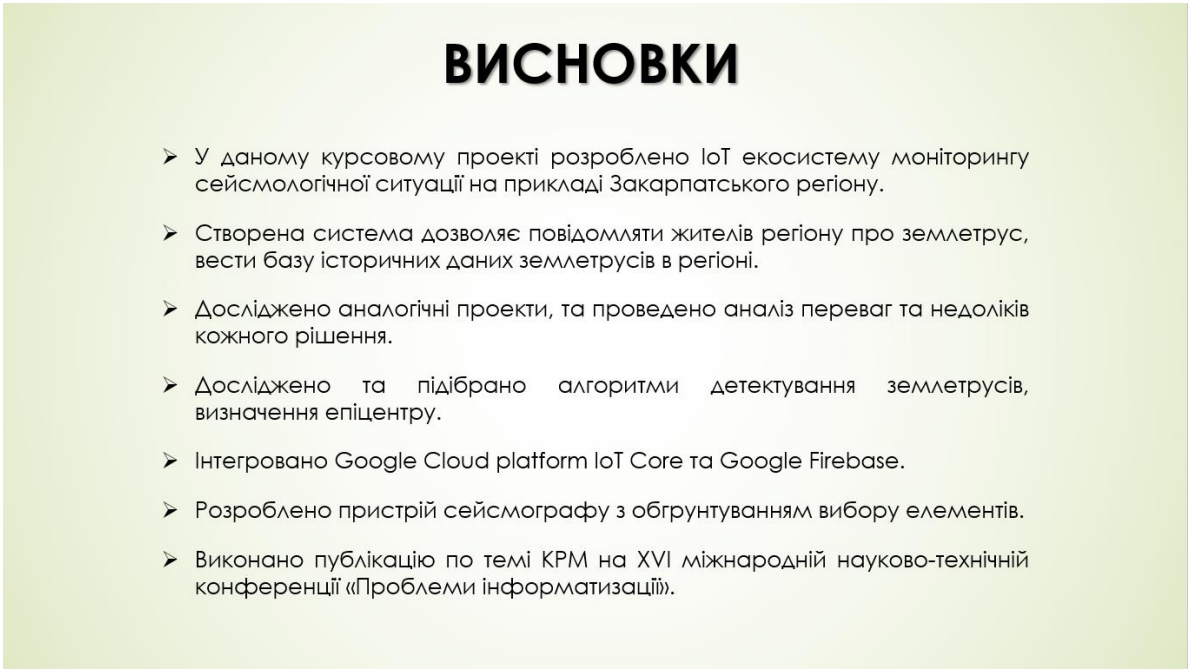

Рисунок В.7 – Слайд 7. Висновки

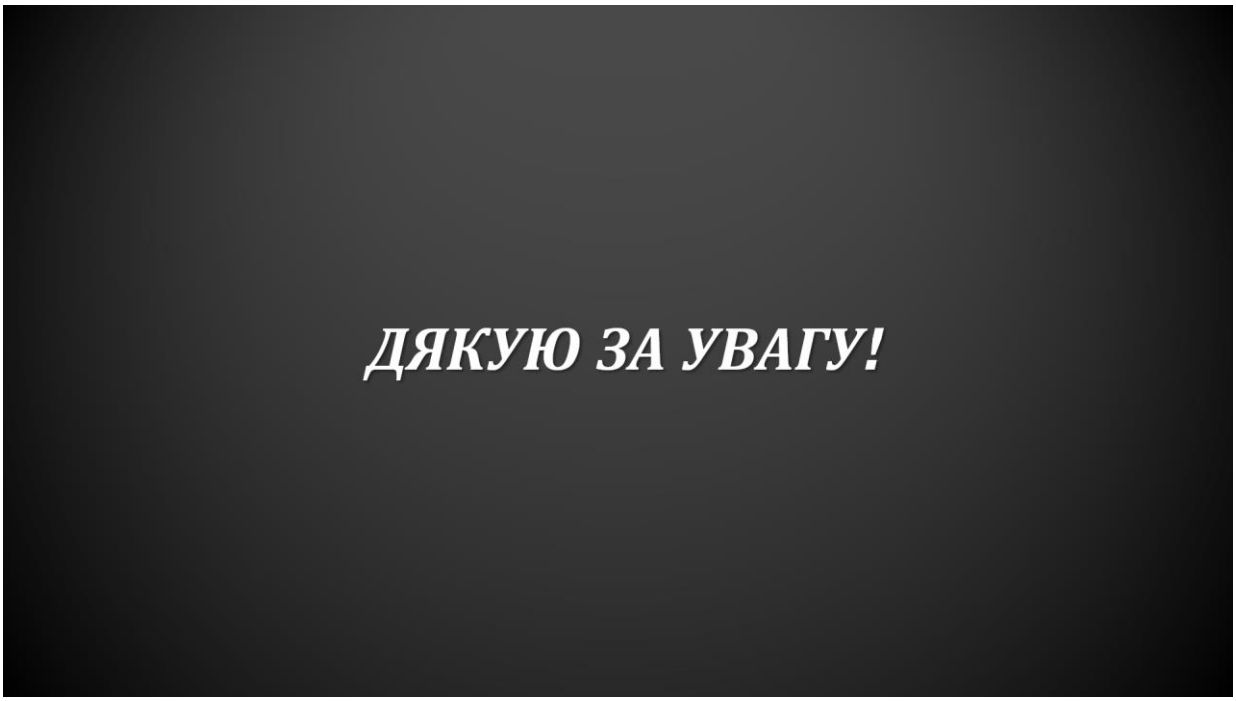

Рисунок В.8 – Слайд 8. Дякую за увагу