Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Програмна система передчасного попередження про рух спеціального транспорту на перехресті»

Виконав: студент 2-го курсу,
групи 2ПІ-20м
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Подобрій В. І
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри ПЗ
Рейда О.М.
(прізвище та ініціали)
«____»_____2021 р.

Опонент:
д.т.н., проф. Василевський О. М
(прізвище та ініціали)
«____»_____2021 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«____»_____2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 13 » вересня 2021 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Подобрію Владиславу Ігоровичу

1. Тема роботи – Програмна система передчасного попередження про рух спеціального транспорту на перехресті.

Керівник роботи: Рейда О. М., к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 24 » вересня 2021 р. № 277.

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи:

Стандартизовані звукові сигнали для систем ПЕТЗ, параметри сигналів систем ПЕТЗ, звукові файли із сигналами систем ПЕТЗ, вимоги до системи визначення звукового сигналу (наявність датчиків звуку по всіх напрямках перехрестя), використання автономної системи реагування на сигнали ПЕТЗ.

4. Зміст розрахунково-пояснювальної записки: вступ, аналіз існуючих методів побудови систем екстрених транспортних засобів, аналіз методів розпізнавання звуку, аналіз програмних засобів додатку, розробка методу аналізу звуку, проектування та розробка системи пріоритету екстрених

транспортних засобів, експериментальні дослідження системи,економічна частина.

5. Перелік графічного матеріалу: актуальність обраної теми; основні задачі дослідження; наукова новизна одержаних результатів; порівняльний аналіз аналогів; архітектура системи; лістинг

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Рейда О. М., к.т.н., доцент кафедри ПЗ		
5	Бурєннікова Н. В., д.е.н., проф. кафедри ЕПВМ		

7. Дата видачі завдання 14 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів розпізнавання звуку та побудови систем екстрених транспортних засобів	15.09.2021-26.09.2021	Вик.
2	Розробка методу розпізнавання звуку	27.09.2021-15.10.2021	Вик.
3	Проектування системи пріоритету екстрених транспортних засобів	16.10.2021-7.11.2021	Вик.
4	Програмна реалізація системи пріоритету екстрених транспортних засобів	8.11.2021-21.11.2021	Вик.
5.	Економічна частина	22.11.2021-30.11.2021	Вик.

Студент _____ **Подобрій В. І.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Рейда О. М.**
(підпис) (прізвище та ініціали)

Опонент магістерської кваліфікаційної роботи _____ **Василевський О.М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 681.326.3

Подобрій В. І.: Програмна система передчасного попередження про рух спеціального транспорту на перехресті. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 134 с.

На укр. мові. Бібліогр.: 33 назв; рис.: 27; табл.: 17.

У магістерській кваліфікаційній роботі проведено детальний аналіз методів і засобів організації систем пріоритету екстрених транспортних засобів, обґрунтовано необхідність підвищення їх продуктивності та ефективності.

Розглянуто та проаналізовано існуючі методи та підходи в задачах розпізнавання звуку. Запропоновано метод розпізнавання звуку сирени. На його основі спроектовано та розроблено систему, що розпізнає екстрені транспортні засоби та керує світлофорами на перехресті.

Проведено тестування системи на різних вхідних даних та доведено її працездатність та ефективність.

Розроблену систему можна використовувати для зменшення часу проїзду екстрених транспортних засобів, що в свою чергу дозволить автомобілям швидкої допомоги, пожежним та поліції швидше діставатися до місця призначення.

У процесі досліджень використовувались: теорія сигналів, теорія диференціально-інтегрального числення, гармонічний аналіз, функціональний аналіз. Комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Ключові слова: розпізнавання звуку, сирена, дорожній рух, спеціальний транспорт, світлофор

ABSTRACT

УДК 681.326.3

Podobriy V. I.: Software system for early warning of crossing emergency vehicles Master's thesis in specialty 121 – software engineering. Vinnitsa: VNTU, 2021. 134 p.

In Ukrainian language. Bibliographer: 33 titles; fig.: 27; tabl.: 17.

In the master's qualification work methods and means of creation of emergency vehicles priority systems is analyzed. Proved necessity of increasing its productivity and efficiency.

The existing methods and approaches in sound recognition problems are reviewed and analyzed. A method of siren sound recognition is proposed. Based on it, a vehicle priority system has been designed and developed. It recognizes emergency vehicles and controls traffic lights at intersections.

The system was tested on various input data and its efficiency and effectiveness were proved.

The developed system can be used to reduce the travel time of emergency vehicles, which in turn will allow ambulances, firefighters and police to get to their destination faster.

In the process of research were used: signal theory, theory of differential-integral calculus, harmonic analysis, functional analysis. Computer modeling for analysis and verification of the obtained theoretical propositions.

Key words: sound recognition, siren, traffic, emergency vehicle, traffic lights

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Управління рухом транспортних засобів відіграє важливу роль у будь-якій інтелектуальній системі керування. Послідовність та тривалість зелених сигналів на перехресті є двома ключовими аспектами, які слід враховувати при керуванні світлофором. У більшості країн світлофори мають фіксовану послідовність та тривалість світла. Однак фіксовані методи керування підходять лише для стабільного та регулярного руху, але не для динамічних ситуацій. З огляду на сучасний стан практики, послідовність зеленого вогню визначається без урахування можливої присутності аварійних автомобілів. Таким чином, спеціальні транспортні засоби, такі як машини швидкої допомоги, поліцейські, пожежні машини тощо, повинні чекати в дорожньому русі на перехресті, що затримує їх прибуття до місця призначення і спричиняє втрату життів та майна.

Зростаюча з кожним роком кількість транспортних засобів на дорогах не тільки збільшує час реагування аварійних машин, але й збільшує шанси їх потрапити в аварії. Автомобіль швидкої допомоги, який виїжджає на перехрестя на великій швидкості на червоне світло, створює небезпеку для руху на інших дорогах і може спричинити аварію. За даними національної поліції України з 2018 по 2021 рік, 321 автомобіль швидкої допомоги став учасником ДТП [1]. До того ж у близько 6% випадків "швидка" приїжджає до місця призначення більше ніж через 20 хвилин, а 2% з усіх викликів закінчуються смертю пацієнта [2]. Враховуючи ці дані можна зробити висновок, що існує серйозна потреба в інтелектуальній системі управління рухом для ефективного керування як звичайними, так і аварійними транспортними засобами.

Існують різні способи реалізації систем перемикання світлофорів, щоб спеціальний транспорт міг проїхати перехрестя в пріоритетному порядку. Зокрема, одна з ідей полягає в тому, щоб розмістити спеціальний передавач на кожному транспортному засобі, а в кожен світлофор додати приймач, що буде приводити в дію механізм перемикання світлофора.

Однак цей метод є відносно дорогим і громіздким, до того ж персонал автомобіля повинен кожен раз вручну приводити передавач у дію.

Інша ідея полягає у використанні світлофорів, оснащених детекторами, що здатні виявляти проблискові вогні (зазвичай спеціальні стробоскопи), встановлені на кожному спеціальному транспортному засобі. Цей варіант має певні переваги у вартості та корисності, оскільки автомобілі швидкої допомоги зазвичай оснащені блимавкою, що вмикається у разі надзвичайної ситуації. Однак перевага у вартості зменшується, якщо для активації детектора, який взаємодіє з контролером світлофора, необхідно передбачити спеціальний тип сигнальних вогнів. Крім того, така система схильна до хибного виявлення, оскільки немає правил щодо використання проблискових вогнів на транспортних засобах, які не є аварійними. Таким чином, приватний транспортний засіб, обладнаний проблисковими маячками, може спричините хибне спрацювання детектора. Крім того хибне спрацювання також можуть викликати рекламні вивіски, комерційні вітрини та декоративне освітлення.

Таким чином кращим підходом до вирішення проблеми може бути розпізнавання звуків, що видають сирени автомобілів спеціального призначення. Цей підхід має явну перевагу в тому, що для його реалізації не потрібно встановлювати спеціальне обладнання на кожен автомобіль. Ще одна перевага полягає у тому, що сирени вмикаються лише у надзвичайних ситуаціях і до того ж правила забороняють використання сирен у не-спеціальному транспорті.

Детектор сирени – це система, що здатна розпізнавати звуки сирени, тобто звуки, миттєві частотні складові яких змінюються з відомою швидкістю в межах вибраного діапазону частот і з відомим періодом. Розпізнавання звуків сирени, які видає спеціальний транспорт, може бути використано для розробки системи ефективного керування світлофором.

Мета та завдання дослідження. Метою роботи є оптимізація режиму роботи світлофорів на перехресті з метою забезпечення пріоритетного проїзду спеціального транспорту: автомобілів швидкої допомоги, поліцейських, пожежних, тощо.

Основними задачами дослідження є:

- Провести аналіз існуючих методів і засобів контролю світлофорів для проїзду спеціального транспорту.
- Запропонувати новий метод розпізнавання наявності спеціального транспорту на перехресті.
- На основі запропонованого методу розробити систему для розпізнавання наявності спеціального транспорту та контролю світлофорів на перехресті.
- Провести експериментальні випробування розробленої системи.

Об’єкт дослідження – процес контролю рухом автотранспорту на перехресті під час перетину його спеціальним автотранспортом у автономному режимі.

Предмет дослідження – методи та засоби розпізнавання наявності спеціального транспорту на перехресті, управління рухом автотранспорту.

Методи дослідження. У процесі досліджень використовувались: теорія сигналів для обробки звукових сигналів, теорія диференціально-інтегрального числення, гармонічний аналіз, функціональний аналіз, методи частотного аналізу для визначення сигналів спеціального призначення. Комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

На основі виконаних теоретичних і експериментальних досліджень вирішено проблему визначення звукових сигналів, оптимізацію руху на перехресті під час руху спецтранспорту, а саме:

– Розроблено класифікацію систем пріоритету екстрених транспортних засобів (ПЕТЗ).
– Подальшого розвитку отримав метод визначення звуку на основі фільтрування та амплітудного аналізу.
– Запропоновано архітектуру системи пріоритетного перетину перехрестя транспортними засобами екстрених служб.

Практична цінність отриманих результатів. На основі розробки, що розглядається в даній роботі, можливо побудувати систему, що здатна значно

зменшити час проїзду екстрених транспортних засобів через перехрестя та збільшити кількість врятованих життів

Структура та обсяг роботи. Відповідно до мети і завдань дослідження структура роботи складається зі вступу, п’яти розділів, висновків, списку використаної літератури та додатків. За час роботи опрацьовано 33 літературних джерела. Зміст роботи викладено на 88 сторінках машинописного тексту.

Джерелами інформації для вирішення перерахованих вище завдань є збірники наукових праць, монографії, періодична література, підручники та довідники, періодичні фахові журнали.

1 ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Порівняльний аналіз систем пріоритету екстрених транспортних засобів

Загальновідомо, що якщо людині, яка постраждала на серцевий напад надати відповідну допомогу протягом першої години після інциденту, у цієї людини збільшуються шанси на виживання і одужання. Ця перша година після серцевого нападу називається золотою годиною. Однією з найпоширеніших причин смерті у випадку ДТП є втрата великої кількості крові. Вагітну жінку, що збирається народити, необхідно швидко транспортувати до лікарні, щоб уникнути певних ускладнень. Пожежним автомобілям необхідно вчасно прибути на місце інциденту, щоб стримати поширення вогню та врятувати людей, що могли потрапити в пастку. Тому автомобілі спеціального призначення повинні мати можливість дістатися до місця якомога швидше. З іншого боку загальна кількість транспортних засобів у містах збільшується в геометричній прогресії. Для забезпечення потреб зростаючого населення, органам місцевого самоврядування доводиться постійно збільшувати кількість громадських транспортних засобів, таких як автобуси, таксі тощо. В той же час більшість великих міст мають дуже жорсткі обмеження щодо простору, а отже, вулиці та дороги дуже вузькі. Всі ці фактори сприяють збільшенню часу в дорозі для всіх транспортних засобів, що становить жахливу проблему для таких автомобілів спеціального призначення. Однак ця проблема не є недавньою, вона існує десятиліттями. Так вже було запропоновано багато інноваційних рішень, а деякі з них реалізовано. Ці рішення можна називати системами ПЕТЗ.

У цьому розділі ми розглянемо існуючі системи ПЕТЗ. Як показано на рис. 1.1, існує п’ять основних типів таких систем, а саме на основі акустики, на основі прямої видимості, на основі GPS, на основі бездротового зв’язку та на основі вузлів [3].

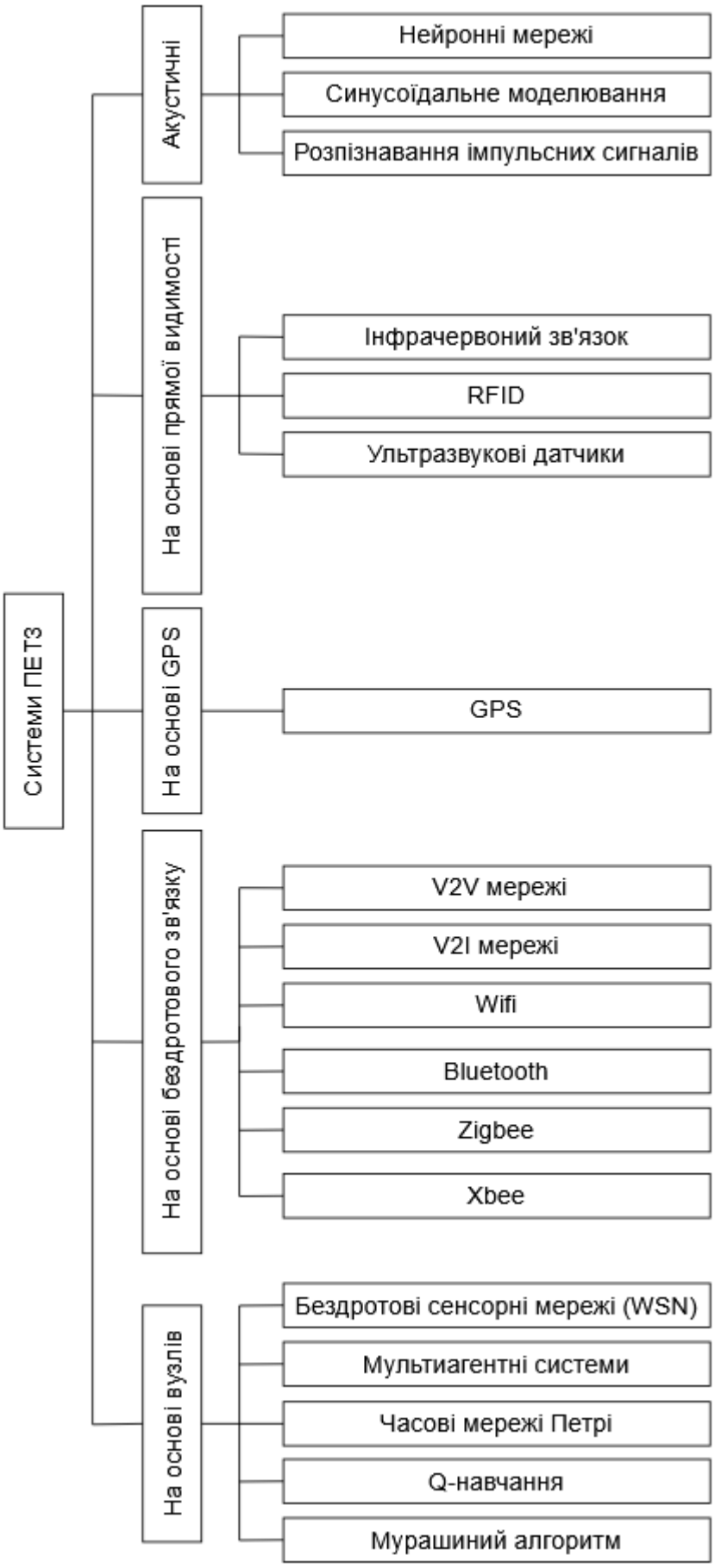


Рисунок 1.1 – Класифікація систем ПЕТЗ

Основна мета систем ПЕТЗ - скоротити час, що проходить з моменту виклику до моменту прибуття автомобіля на місце призначення. Це може досягатися шляхом створення «коридору» на дорозі, шляхом управління транспортним потоком на перехрестях, шляхом розрахунку найбільш оптимального шляху тощо.

Загальний робочий процес будь-якої системи ПЕТЗ складається з виявлення транспортного засобу, збору інформації, аналізу інформації та, нарешті, запуску пріоритетної стратегії. Інформація про автомобіль може включати в себе його поточне місцезнаходження, швидкість, курс, пункт призначення, маршрут, яким він планує їхати тощо. Пріоритетна стратегія може включати в себе розрахунок оптимального маршруту, керування рухом на перехрестях, надання даних про спеціальний транспортний засіб іншим учасникам дорожнього руху тощо.

Надалі ми розглянемо кожен тип систем ПЕТЗ відповідно до класифікації, представленої на рис. 1.1.

Транспортні засоби спеціального призначення можна виявити за допомогою різних методів, таких як обробка зображень, бездротові приймачі та передавачі, радіочастотна ідентифікація (RFID), глобальна система позиціонування (GPS) тощо. Але всі ці методи вимагають певної модифікації самого автомобіля: оснащення бездротовими передавачами, RFID-мітками та GPS модулями. Для того, щоб використовувати обробку зображень, необхідні високоякісні камери разом з процесорами високої обчислювальної потужності, що знову ж таки підвищую загальну вартість системи. З іншого боку, ми знаємо, що всі автомобілі спеціального призначення використовують сирени. Тож, використовуючи методи обробки звуку, ми можемо виявляти такі автомобілі та відповідно реагувати, при цьому не має необхідності додатково модифікувати кожен автомобіль.

В. Тран та ін. [4] пропонують модель на основі згорткової нейронної мережі (CNN) під назвою SirenNet для розпізнавання та класифікації сирен автомобілів швидкої допомоги, пожежних та поліції. Запропонована модель

складається з двох частин, за допомогою яких обробляються дані і робляться прогнози. Одна з частин використовує двовимірну модель CNN під назвою MLNet, а інша використовує одновимірну модель CNN під назвою WaveNet. Передбачення цих двох моделей усереднюються для визначення остаточного прогнозу. Таку архітектуру називають ансамблевою архітектурою (рис 1.2). В ансамблевій архітектурі використовуються дві або більше моделей, де кожна модель отримує однакові вхідні дані, а вихідні дані агрегуються для визначення результату.

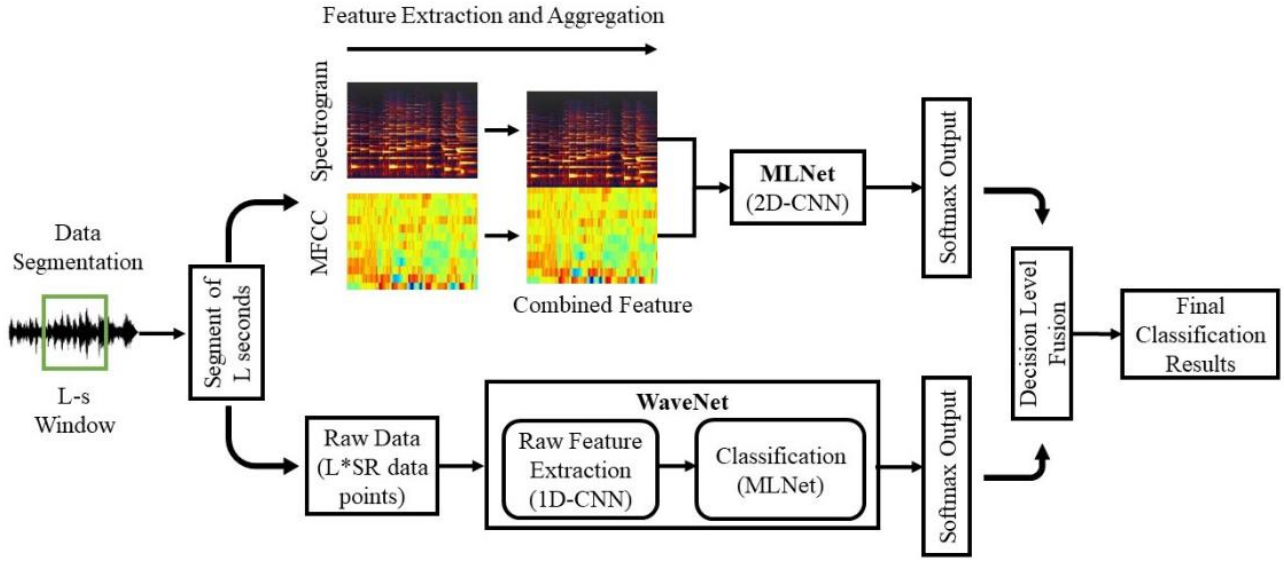


Рисунок 1.2 – Ансамблева архітектура призначена для розпізнавання та класифікації сирен транспорту спеціального призначення.

Дослідники зібрали набір аудіо даних, що складається зі звуків швидкої допомоги, поліції, сирен пожежних автомобілів а також шуму дорожнього руху з різних джерел. Загалом їх набір даних складався з 8742 звуків сирен, 7552 автомобільних гудків та 10381 звуків міського шуму. Потім автори розділили весь набір даних на п’ять рівних частин, щоб уникнути переважання звуків одного типу під час навчання двох базових моделей. Варто зазначити, що MLNet і WaveNet використовують кардинально різні способи для виявлення потрібних звуків. MLNet використовує кепстральні коефіцієнти і логарифмічні

спектрограми, у той час як WaveNet використовує необроблені одновимірні аудіо-дані.

Незважаючи на те, що в цьому розділі ми обговорюємо акустичні системи пріоритету транспортних засобів екстреної допомоги, цікаво зрозуміти і інші системи, що використовуються у схожих задачах. Наприклад прилади для людей з вадами слуху. Елліс Д. П. та ін. [5] пропонують систему, яка може виявляти не тільки сирени, а й сигнали тривоги всіх видів, як-то пожежну сигналізацію, димову сигналізацію, сигналізацію системи загального попередження а також звичайні у побуті звуки: дверний дзвінок, дзвінок телефона, звуки, що видають домашні тварини тощо. Основна увага в цій роботі зосереджена на виявленні попереджуючих сигналів у загальному сенсі, тобто не тільки тих, на яких система проходила навчання. Автори стверджують, що більшість підходів до розпізнавання таких звуків полягають у визначенні та аналізі загальних характеристик аудіо-даних, але всі такі підходи дуже схильні до впливу стороннього шуму. У своїй роботі автори пропонують альтернативний підхід до розпізнавання звуків, він полягає у тому, щоб спочатку отримати з аудіо-файлу всі унікальні акустичні події, а потім на основі різних властивостей цих подій проводити класифікацію. За словами авторів, такий підхід менше залежить від шуму, оскільки аналізуються окремі акустичні події, а не весь файл в цілому.

Загалом авторами було розроблено дві системи виявлення звуків, одна з яких базується на звичайному підході з використанням нейронної мережі (рис 1.3, а), а інша використовує вищеописаний підхід, що ділить аудіо-дані на унікальні акустичні події і намагається відділити потрібні звуки від фонового шуму (рис. 1.3, б).

Технології дозволяють людям з обмеженими можливостями вести краще життя та робити свій внесок до суспільства. До прикладу, такі люди отримують можливість працювати на рівні з іншими а також добиратися до місця роботи на своїх автомобілях. Але, говорячи про водіння, для людей з вадами слуху все ще може бути складно виявляти сирени автомобілів спеціального призначення, що наближається, поки він не під’їде дуже близько.

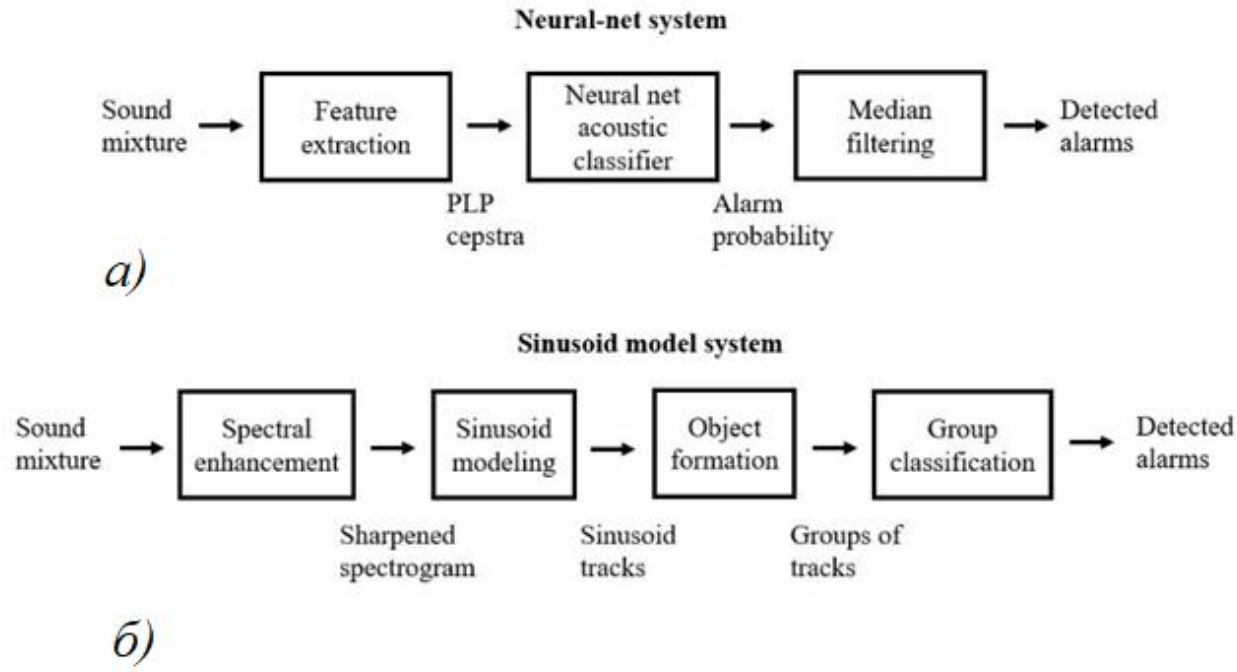


Рисунок 1.3 – Запропоновані Елліс Д. П та ін. підходи до вирішення проблеми розпізнавання та класифікації звуків: а – система, що базується на нейронній мережі, б – система синусоїдальної моделі

Для вирішення цієї проблеми Бертеллі та ін. [6]. запропонували систему, що може виявляти сирени автомобілів і надавати інформацію про них водію. Система представляє собою нейронну мережу, в яку подаються певні особливості вхідного аудіо. Для навчання та тестування системи використовувалася база аудіо-даних, що були записані в Італії в різних умовах, включаючи запис зсередини та ззовні автомобіля на швидкості і без, під час різної погоди, з різних відстаней тощо.

На першому етапі кожен вхідний звук поділявся на кадри по 10 мс, для кожного з яких було розраховано 12 кепстральних коефіцієнтів (MFCC). Після цього результат був переданий на вхід нейронної мережі, що має 12 вхідних нейронів, 24 нейрони прихованого шару та 2 виходи. Нейронна мережа була натренована за допомогою алгориму зворотнього розповсюдження.

Автори зазначають, що кадри більшої тривалості, будуть давати більш якісні результати, але це зменшує загальну швидкодію системи, тому було вирішено зберегти тривалість кадру 10 мс, але додатково обробляти вихідні дані

нейронної мережі. Кожні 20 мс. вихідних даних обробляються шляхом усереднення. Система була протестована і було визначено, що вона може ідентифікувати сирену по першим 400 мс. аудіо-даних з точністю 99%.

Одним із застосувань систем для виявлення сирен та сигналів тривоги є виробництво. У багатьох виробничих підприємствах працівники повинні носити навушники із шумопоглинанням, що захистити свої вуха від звуків розташованих поблизу машин. Хоча ці навушники захищають слух працівників, вони також можуть викликати проблеми під час надзвичайних ситуації тому, що не дають змогу почути сигнали аварійної сирени. В свою чергу, неможливість розпізнати надзвичайну ситуацію з боку працівників може спричинити серйозну затримку в евакуації. М. А. Карбоно та ін. розробили систему виявлення аварійної сирени [7], яка може бути вбудована у малопотужні системи, інтегровані з засобами захисту органів слуху працівників. Такий пристрій може сповіщати працівників про активацію аварійної сирени через їхні навушники.

Алгоритм, запропонований розробниками, оптимізовано для виявлення імпульсних сигналів тривоги, що є найбільш поширеними за заводах. Імульсний сигнал тривоги складається із звукового сигнала, за яким слідує тиша після чого все повторюється. Запропонований алгоритм був перевірений на зразках імпульсної сирени у поєднанні з фоновими шумами, що були взяти з бази даних NOISEX-92. Алгоритм складається з чотирьох основних етапів:

- Смуговий фільтр;
- Виявлення огинаючої;
- Автоматична кореляція;
- Прийняття рішення.

Вхідний аудіофайл спочатку проходить через смуговий фільтр, що пропускає лише частоти в діапазоні від 500 Гц до 1500 Гц. Після цього сигнал пропускається через детектор огинаючої, що генерує сигнал, відповідний рівню амплітуди вхідного сигналу. Для виявлення періодичності над сигналом проводиться автокореляція, після чого запускається алгоритм, що визначає чи є сигнал тривожним: ритми накладаються на відстань між нулями на виході

автокореляції. Оскільки імпульсні тривоги є періодичними, перетини нуля розміщені рівномірно. Під час тестів система показала точність 95%.

Цікаву ідею про те, як можна виявити звук сирени запропонували С. Клонц та ін [8]. У своєму дослідженні вони розділили завдання виявлення тривоги на дві частини. По-перше це механізм фільтрації, по-друге – схема компаратора, в якій через певну затримку активується розпізнавання звуку. В рамках роботи було проаналізовано АЧХ. Фільтр, що використовувався, був отриманий шляхом каскадного поєднання фільтра високих частот з фільтром низьких частот. На рис. 1.4 показано АЧХ розробленого фільтру з нижньою і верхньою частотами зрізу на 1550 і 5550 Гц відповідно. Отриманий графік частотної характеристики було проаналізовано для того, щоб зробити систему більш точною та ефективною.

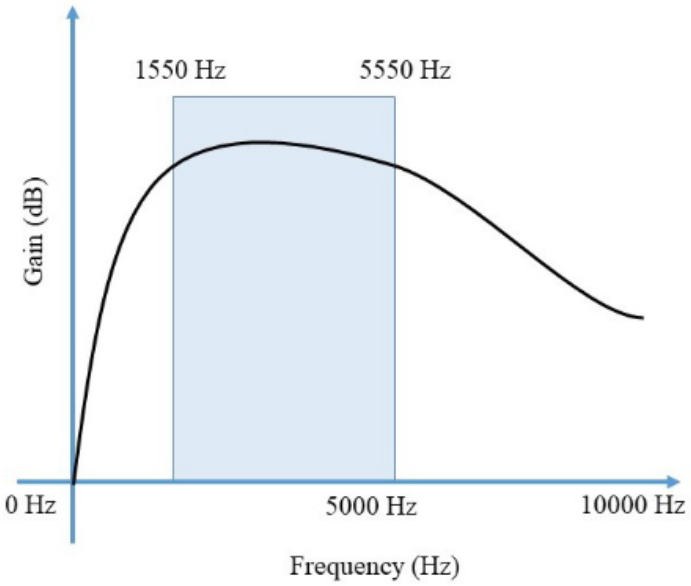


Рисунок 1.4 – АЧХ фільтра

Чудову техніку візуальної індикації сигналу тривоги для людей з вадами слуху, було представлено Д. Кермел та ін. [9], як показано на рис. 1.5. Вони розділили свою систему на дві частини. Перша – це частина, що використовується для навчання моделі класифікатора за допомогою попередньо сформованого набору даних. Друга - класифікація в реальному часі, в рамках

якої нові звуки ідентифікуються з допомогою моделі, отриманої на першому етапі.

Розробники тестували свою систему на наборі даних, що складався з декількох десятків звукових сигналів тривоги та інших сторонніх звуків. Під час тестів система показала точніть близько 98%.

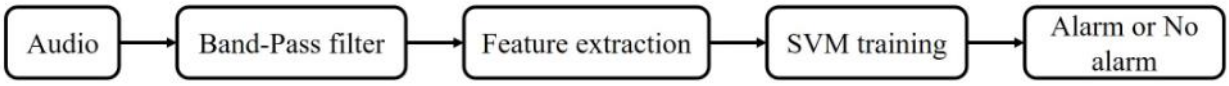


Рисунок 1.5 – Схема системи Д. Кермел та ін.

Меуччі та ін. [11] запропонували систему, що базується на спектральній складовій високої та низької гармонік. Системи на основі швидкого перетворення Фур'є (FFT) або віконного перетворення Фур'є (STFT) не правильно визначають домінантну частоту сирени, оскільки гармоніки мають велике значення в сигналі. Тому дослідники запропонували використовувати фільтрацію з множинними затримками (MDF). MDF – це техніка в часовому просторі, що використовується для класифікації кожної частини аудіосигналу, як тональної або атональної і, шляхом пошуку піків, визначення висоти звуку. Вихідні дані показують наявність або відсутність сирени.

Ще одна система для попередження водіїв про наближення спеціального транспорту включає в себе пристрій формування звукового сигналу, встановлений на автомобілі швидкої допомоги та системи з детектору та пристрою для відображення, що встановлюється на звичайний автомобіль. Ця система була запропонована У. Е. Брілл та ін. [10].

Системи ПЕТЗ на основі прямої видимості використовують такі технології, як радіо-частотна ідентифікація (RFID), інфрачервоні датчики, оптичні сенсори тощо. У будь-якій системі прямої видимості є передавач, що надсилає сигнал та приймач, що його обробляє. Зазвичай передавач

встановлюється на автомобіль спеціального призначення, а приймач або декілька приймачів на перехресті, але існують і інші варіанти.

Як показано на рис. 1.6, С. Шибуя та ін. [12] запропонували систему, яка спрямована на допомогу автомобілям спеіального призначення якомога швидше дістатися до місця призначення за допомогою декількох підсистем, таких як інфрачервоні маяки, вибір найкращого маршруту, організація пріоритетного проїзду на перехрестях тощо. Для реалізації такої системи необхідно обладнати всі транспортні засоби інфрачервоними приймачами та передавачами, а також пристроєм для відображення оптимального маршруту. Фундаментальним блоком запропонованої системи є мережа апаратних вузлів, що здатні забезпечувати двосторонній інфрачервоний зв'язок. Ці вузли, які також називаються «маяками», повинні мати можливість бездротового підключення до командного центру. Такі маяки розміщуються по всьому місту і особливо на дорогах, що прилягають до перехрестя.

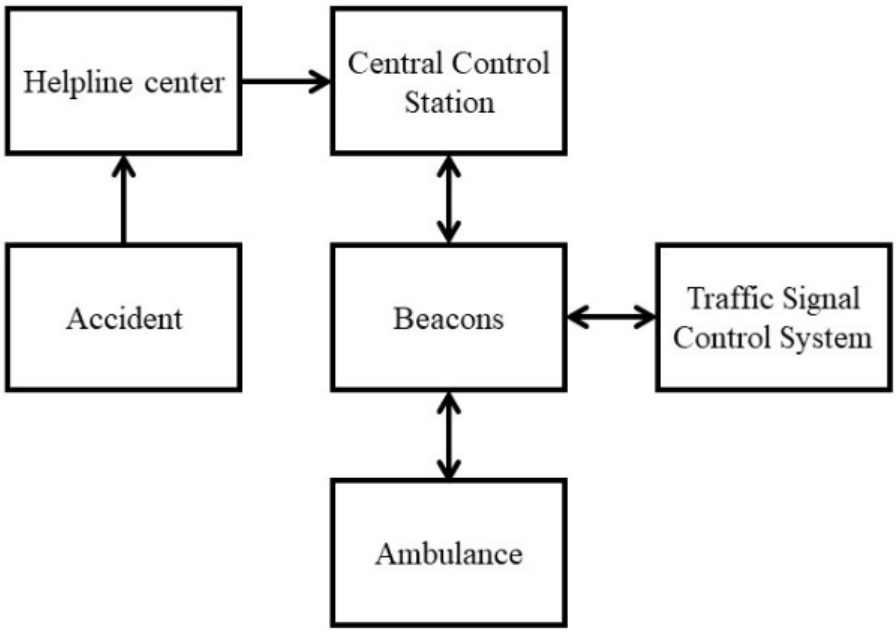


Рисунок 1.6 – Діаграма системи С. Шибуя та ін.

Розглянемо приклад: пацієнт потребує невідкладної медичної допомоги, тому викликають швидку медичну допомогу. З сусідньої лікарні відправляють

автомобіль. Відомості про місцезаходження хворого та ідентифікаційний номер автомобіля швидко надсилаються в командний центр, звідки ця інформація предається на всі маяки. Щоразу, коли маяк отримує інфрачервоний сигнал від автомобіля швидкої допомоги, він витягує з нього ідентифікатор транспортного засобу і звіряє його з ідентифікатором, отриманим з командного центру. Якщо вони збігаються, ця інформація передається в командний центр. Тепер місцерозташування швидкої відоме і можна розрахувати найшвидший шлях до місця розташування пацієнта. Ця інформація надсилається до всіх маяків, що знаходяться на оптимальному шляху, що надалі допомагає організувати систему пріоритетного проїзду швидкої на перехресті.

Коли автомобіль швидкої допомоги наближається до перехрестя, тоді найближчий маяк посилає сигнал в систему управління рухом на перехресті. Залежно від поточного стану сигналів світлофора, система керування або збільшує тривалість зеленого світла або зменшує тривалість червоного. Коли автомобіль успішно проїжджає перехрестя, наступний маяк знову надсилає сигнал в систему управління і світлофори повертаються у звичайний режим роботи. Як тільки швидка забирає пацієнта, розраховується найшвидший шлях назад до лікарні і весь процес повторюється.

Запропонована система була випробувана на деяких перехрестях Токіо та префектури Чіба в 1999 році. Для оцінки ефективності своєї системи автори спочатку спостерігали за часом, за який автомобілі швидкої проїжджають через не обладнані перехрестя, а потім, встановивши необхідне обладнання, порівнювали час. За даними авторів, час у дорозі для автомобілів спеціального призначення скоротився на 12% та 14% у префектурі Тіба та Токіо відповідно.

Р. Бхарадвадж та ін. [13] запропонували систему ПЕТЗ, що базується на RFID-зчитувачах. Запропонована система складається з трьох частин: пристрій для управління дорожнім рухом на перехресті, RFID-зчитувач та пристрій, що рахує кількість машин на дорозі. Останній представляє собою індуктивну петлю, що розташовується під дорогою. Кожен раз, коли через неї проїжджає автомобіль, лічильник збільшується на одиницю. Ці дані по кожній лінії

перехрестя відправляються в систему керування перехрестям і таким чином є дані, скільки автомобілів знаходиться на перехресті у кожен окремий момент часу. Задача RFID-зчитувача – розпізнати транспорт спеціального призначення, який має бути обладнаний RFID-міткою. Коли такий автомобіль проїжджає повз детектор, його унікальний ідентифікаційний номер буде зчитано. Знаючи загальну кількість автомобілів та кількість автомобілів спеціального призначення, система може оптимізувати рух на перехресті. Наприклад, якщо з однієї сторони дві швидких, а з іншої одна, система спочатку дасть 60 секунд зеленого світла для першої сторони, а потім 60 секунд для другої.

Як показано на рис. 1.7, А. Мазум та ін. [14] запропонували схему управління дорожнім рухом з використанням інтернету речей (IOT). Дослідники описали свою ідею наступним чином. Трафік фіксується ультразвуковими датчиками. Ці дані передаються через Wifi та зберігаються у хмарному сховищі. Після цього дані аналізуються і система управління світлоформи вмикає зелене світло для автомобіля швидкої допомоги. Але якщо інший транспорт намагатиметься проїхати попереду, він буде розпізнаний за допомогою RFID-мітки і водій отримає штраф через мобільний додаток.

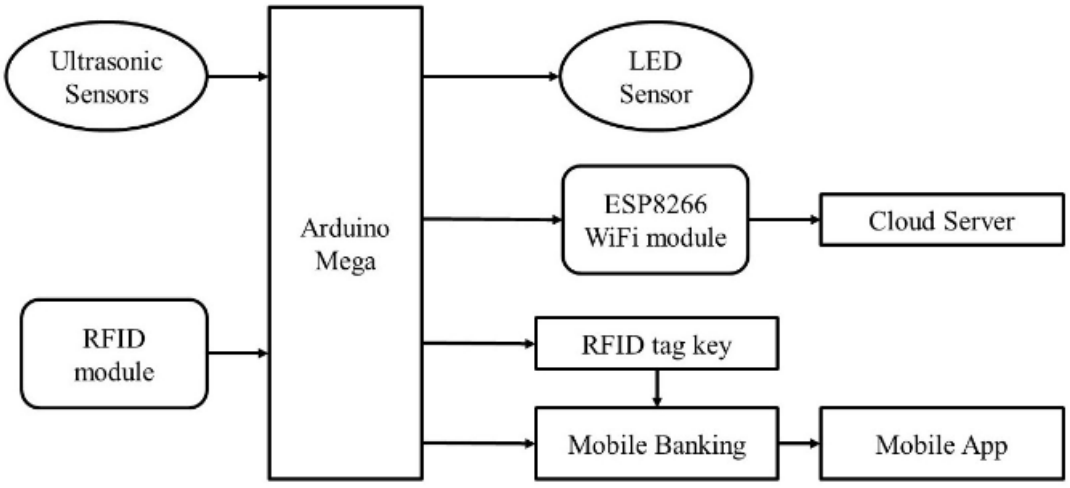


Рисунок 1.7 – Діаграма системи А. Мазум та ін.

Глобальна система позиціонування (GPS) має велику кількість варіантів застосування. Зокрема, вона може бути застосована для реалізації систем ПЕТЗ.

Одну з таких реалізацій запропонували В. Кодайр та ін. [15]. Система використовує GPS для відслідковування поточного місцезнаходження автомобіля швидкої допомоги і його напряму руху. Ця інформація передається в пристрій, що може керувати світлофорами на перехресті. Використовуючи отримані дані, цей пристрій подовжує тривалість зеленого сигналу світлофора на відповідній дорозі і подовжує тривалість червоного сигналу на всіх інших дорогах.

Система складається з двох частин. Перша відповідає за обробку даних GPS та їх відправку через передатчик Xbee. Друга частина складається з отримувача Xbee та мікропроцесора для керування світлофорами. Вона отримує GPS дані, обробляє їх та робить відповідні зміни в роботі світлофорів.

В залежності від рівня заторів на перехресті може використовуватись Xbee модуль короткої чи великої дальності. Також в залежності від рівня заторів змінюється функціональний поріг: тільки якщо відстань від автомобіля швидкої допомоги до перехрестя менша, ніж це значення, роботу світлофорів буде скориговано.

Ще одним прикладом системи, що базується на GPS є система, запропонована Е. С. Елтаіб та ін. [16]. Принцип її роботи схожий на попередню систему, але в цьому випадку використовується мобільна комунікація (GSM) замість Xbee модуля. Ця система складається з трьох основних компонентів: пристрій, що розташовується безпосередньо на автомобілі, пристрій для управління світлофорами та лічильник автомобілів. Частина, що знаходиться в автомобілі складається з GPS модуля і SIM-карти приєднаної до GSM модуля. Відповідно, комунікація відбувається за допомогою SMS-повідомлень. Інший GSM модуль з SIM-картою знаходиться в пристрої керування світлофорами. Лічильник автомобілів розташовується на кожній дорозі, що прилягає до

перехрестя. Принцип його роботи полягає у розпізнаванні тиску, таким чином коли зверху проїжджає автомобіль, значення лічильника збільшується на один.

Коли в службу швидкої допомоги поступає виклик, водій автомобіля вводить адресу призначення в систему, після чого прораховується найкоротший шлях. В автомобіль швидкої допомоги надсилається інформація про всі перехрестя, які знаходяться на оптимальному шляху. Автомобіль починає розсилати дані про своє місцезнаходження на SIM-карти всіх пристроїв керування перехрестям. Використовуючи дані про автомобіль, його швидкість, напрямок, орієнтовний час проїзду через перехрестя, а також поточну ситуацію на перехресті, кількість автомобілів тощо, ці пристрої розраховують оптимальну стратегію керування світлофорами. Як тільки швидка проїжджає перехрестя, пристрій керування перестає отримувати дані про автомобіль і повертає перехрестя в робочий режим.

Системи ПЕТЗ використовують такі технології, як WiFi, Bluetooth, ZigBee, Xbee, GSM та ін. для того, що організувати оптимальну стратегію проїзду автомобілів до місця призначення.

А. Бухеншайт та ін. [17] запропонували систему, що оповіщає всі транспортні засоби, які знаходяться поблизу автомобіля швидкої допомоги про його поточне місцезнаходження, швидкість та напрямок руху. Таким чином водії можуть виконати відповідні дії для забезпечення пріоритетного руху автомобіля спеціального призначення. На додаток до цього, сповіщення також отримує система контролю перехрестя, до якого наближається швидка. В своїй роботі дана система базується на V2I та V2V мережах (VANET), що дозволяє швидко обмінюватись інформацією між транспортними засобами.

У своїй роботі дослідники роблять акцент на загальному баченні системи, а не на використанні конкретних технологій. Система передбачає, що в кожному автомобілі спеціального призначення повинен бути встановлений передавач, а в кожному звичайному автомобілі і в системах контролю перехрестями повинен бути встановлений отримувач. Завдання передавача зібрати всю критично важливу інформацію про автомобіль таку як: запланований маршрут, швидкість,

напрямок, тип автомобіля (поліцейський, швидкої допомоги, пожежний), розмір (маленький, середній великий) тощо. Ця інформація повинна постійно оновлюватися, поки автомобіль їде до місця призначення. Передавач постійно розсилає цю інформацію всім отримувачам, що знаходяться у радіусі його дії або ж тим, що знаходяться на запланованому маршруті. Водій автомобіля спеціального призначення може на ходу змінювати маршрут, в залежності від кількості заторів, ситуації на дорогах, ремонтних робіт тощо і інформація про це буде оновлюватися та розсилатися в режимі реального часу.

А. Бухеншайт та ін. [17] не дають в своїй роботі ніяких точних деталей про пристрої отримувача та передавача або про те, як саме вони повинні обмінюватися інформацією, натомість вони фокусуються на загальному підході до проблеми, демонструючи нам своє бачення того, як її можна вирішити.

Система була протестована на платформі для симуляції транспортних ad-hoc мереж U2VAS. Результати тестів були визнані авторами успішними.

М. Б. Юнес та ін. [18] розробили систему контролю сигналами світлофорів, що аналізує транспортний потік на кожній дорозі, прилягаючій до перехрестя, для того, щоб визначити послідовність сигналів світлофора.

На початку роботи, автори проаналізували транспортний потік на перехресті. Автомобіль, що під’їжджає до перехрестя має вибір: повернути направо, наліво або поїхати прямо. Як показано на рис. 1.8, для правосторонньої системи водіння можна з упевненістю припустити, що транспортний засіб може повернути праворуч на перехресті, не заважаючи іншим учасникам дорожнього руху. Таким чином кожен транспортний засіб повинен покладатися на сигнали світлофора у разі, якщо хоче повернути наліво або поїхати прямо.

Проаналізувавши щільність транспорту на кожній прилягаючій до перехрестя дорозі, можна задати послідовність пар не конфліктуючих між собою транспортних потоків і таким чином значно зменшити навантаженість на перехресті, як показано на рис. 1.9.

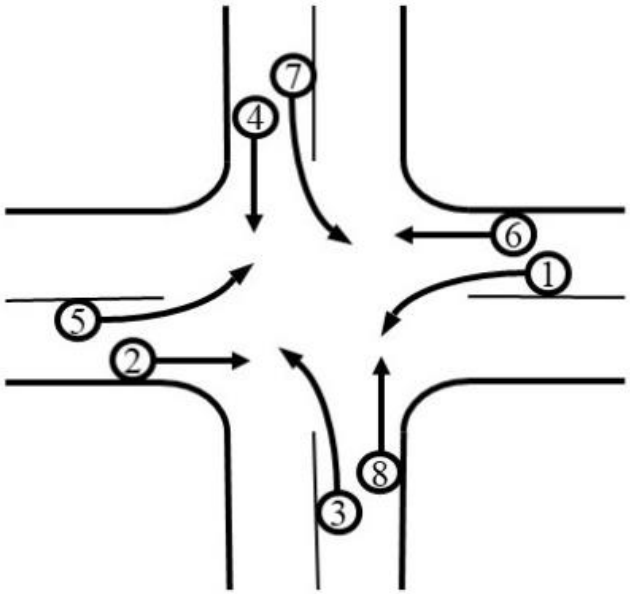


Рисунок 1.8 – Схема руху автотраспорту на перехресті.

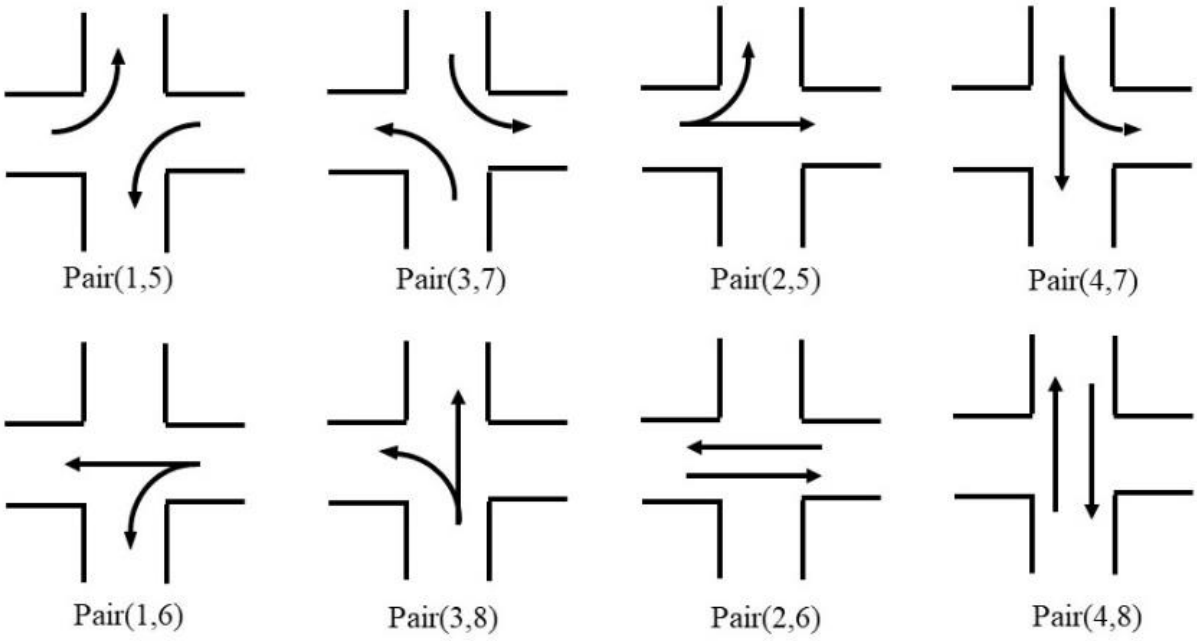


Рисунок 1.9 – Вісім пар, не конфліктуючих між собою транспортних потоків.

Для того, щоб вирішити, яка з пар повинна бути увімкнена у кожен окремий момент часу, необхідно проаналізувати щільність транспортних засобів і знайти транспортний потік з найбільшою щільністю. Оскільки кожен

транспортний потік пов’язаний з двома іншими, вибираємо з них також потік з найбільшою щільністю і таким чином формуємо пару транспортних потоків, що має бути увімкнена на фіксований проміжок часу. Після цього алгоритм вибору пари транспортних потоків повторюється, поки всі пари не буде визначено, не враховуючи вже визначені пари. В результаті отримуємо чотири пари транспортних потоків, які вмикаються по черзі, після чого все повторюється з початку.

Коли екстрений транспортний засіб під’їжджає до перехрестя, система маркує транспортний потік, в якому він знаходиться, як найбільш пріоритетний. При розрахунку нової послідовності, пріоритетний потік буде завжди в першій парі, не зважаючи на щільність транспортних засобів. У разі, якщо декілька екстрених транспортних засобів під’їжджають до перехрестя, система покладається на додаткову інформацію про ці транспортні засоби: ступінь терміновості, відстань до місця призначення тощо. Також система враховує тип автомобіля: найбільш пріоритетними є автомобілі швидкої допомоги, потім пожежні і поліцейські.

А. Шаабан та ін. [19] запропонували систему, що спрямована на зменшення часу проїзду екстрених автомобілів, але разом із тим і на мінімізацію впливу на дорожній рух в цілому.

Робота системи складається з двох основних фаз. Перша – розрахунок найкоротшого шляху до місця призначення. Друга – організація пріоритетного проїзду.

Під час другої фази розраховується певне порогове значення відстані на основі інформації про напрямок та швидкість руху автомобіля та час, що необхідно витратити на зміну сигналів світлофорів. Коли відстань від автомобіля стає менше, ніж це порогове значення, система контролю світлофорів припиняє нормальний режим роботи, натомість дозволяється тільки рух зі сторони автомобіля швидкої допомоги, потім нормальний режим роботи відновлюється. Мінімізація впливу на дорожній рух досягається саме

використанням порогового значення, оскільки світлофори перемикаються тільки коли автомобіль швидкої допомоги під’їде достатньо близько до перехрестя.

К. Байгрейв та ін. [20], запропонували метод сповіщення водіїв про наявність поряд екстреного транспортного засобу. Сигнали короткої дальності наприклад WiFi, Bluetooth, IEEE 802.11 (ультра високочастотний діапазон) можуть бути використані для передачі інформації від автомобіля спеціального призначення в найближчий автомобіль, який в свою чергу надішле інформацію до наступного автомобіля, як показано на рис. 1.10.

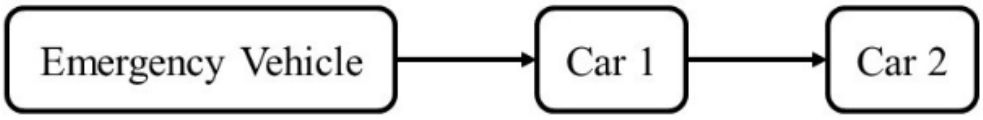


Рисунок 1.10 – Система К. Байгрейв та ін.

Як показано на рис. 1.11, Р. Сундар та ін. [21] опублікували роботу, в якій частина присвячена темі пріоритетного проїзду екстрених транспортних засобів. Для цього в них використовується модуль Zigbee та мікроконтролер. Автомобіль швидкої допомоги посилає сигнал, що складається з унікального ідентифікаційного коду та коду безпеки. Система управління світлофорами на перехресті отримує сигнал, порівнює код безпеки з тими, що є в базі даних і вмикає зелений сигнал для швидкої допомоги, а для всіх інших червоний.

Системи пріоритету екстрених транспортних засобів на основі вузлів використовують такі технології, як бездротові сенсорні мережі (WSN), мульти-агентні системи та ін. для того, щоб розпізнати транспортний засіб, ініціювати та виконати оптимальну стратегію забезпечення пріоритетного проїзду.

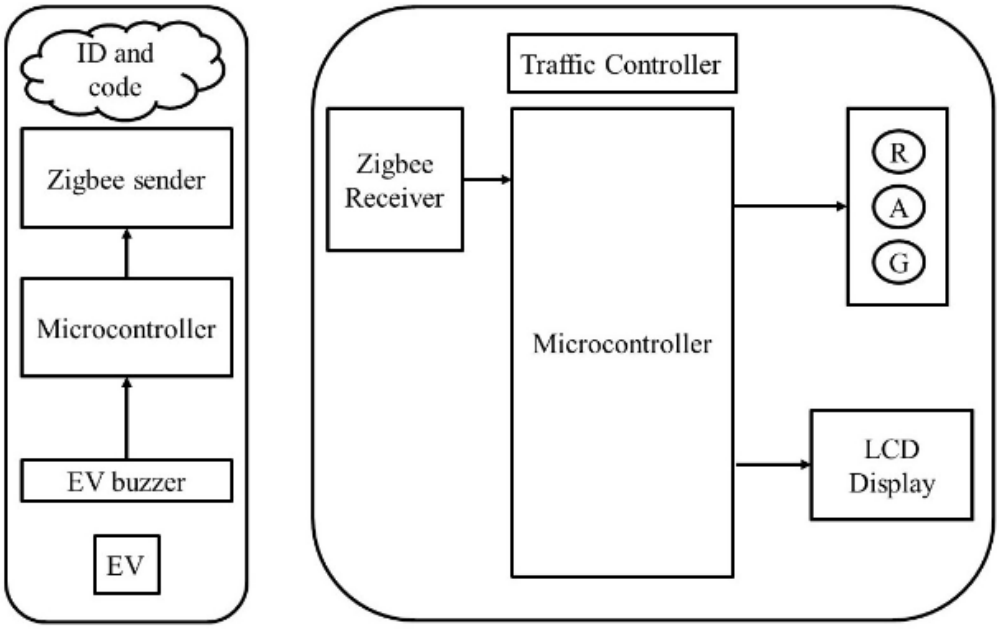


Рисунок 1.11 – Діаграма системи Р. Сундар та ін.

М. Масуд та ін. [22] розробили та протестували систему пріоритету екстрених транспортних засобів, що дозволяє автомобілям мінімізувати час проїзду через перехрестя. Запропонована система базується на контролі сигналів світлофора для того, щоб надати право пріоритетного проїзду спеціальному транспорту. Система використовує бездротову сенсорну мережу для розпізнавання автомобілів швидкої допомоги і контролю світлофорів.

Алгоритм, що розробили автори працює на основі протоколу дерева колекцій (CTP). Щоб зрозуміти запропоновану систему ми повинні взяти за приклад стандартне чотирьох-стороннє перехрестя. На кожній з чотирьох сторін на деякій відстані від перехрестя розміщують вузол WSN. В середині перехрестя також розміщується WSN вузол, який називається базова станція (БС). Чотири вузла, що розміщені на дорогах повинні бути на однаковій відстані від базової станції і так, щоб тільки базова станція була в зоні досяжності кожного з них, але не інші вузли.

Припустимо, що автомобіль швидкої допомоги оснащений передавачем короткої дистанції а всі вузли оснащені відповідними отримувачами. Коли автомобіль швидкої допомоги під’їжджає до перехрестя по одній із чотирьох

доріг, WSN вузол встановлений на цій дорозі отримує його сигнал і інформує базову станцію про наближення швидкої допомоги. Базова станція вмикає для автомобіля швидкої допомоги зелений сигнал світлофора, а для всіх інших - червоний сигнал. Коли автомобіль проїжджає перехрестя, наступний вузол WSN інформує про це базову станцію і вона повертає світлофори в нормальний режим роботи.

Всі комунікації між WSN вузлами і базовою станцією працюють на основі розробленого авторами алгоритму WSN-EVP (emergency vehicle priority). Алгоритм працює на основі топології дерева, де базова станція – кореневий вузол, а інші чотири – листові вузли. Автомобіль швидкої – дочірній вузол WSN вузла, що його «помітив». Алгоритми на основі CTP мають три основних компонента і кожен вузол представляє в мережі всі три. Перший компонент називається рушій маршрутизації (Routing Engine, RT), що оновлює таблицю маршрутизації, коли змінюється топологія мережі. Таблиця маршрутизації кожного вузла зберігає інформацію про очікувану кількість передач (Expected Transmission Count, ETX) інших вузлів, що знаходяться в його зоні досяжності. ETX метрика показує, як швидко можна досягнути базову станцію з певного вузла. Рушій маршрутизації допомагає вузлу обрати батьківський вузол, через який він може послати повідомлення до базової станції якомога швидше. Другий компонент – це оцінювач зв’язку (Link Estimator, LE). Він відслідковує надійність підключення до кожного вузла, що знаходяться в зоні досяжності. За допомогою цього компонента вузол може змінювати його батьківський вузол, якщо бачить, що в зоні його досяжності є вузли з більш стійким з’єднанням. Останній компонент – це рушій пересилання (Forwarding Engine, FE). Він використовується, щоб посилати дані на інший вузол. В даному випадку, він потрібен, щоб швидка могла послати дані на базову станцію через її батьківський вузол.

Ю. С. Хуанг [23] та ін. запропонували використовувати часові мережі Петрі (Timed Petri Nets, TPN). Часові мережі Петрі - це один з інструментів математичного моделювання, як то, наприклад UML. TPN складається з вузлів,

переходів та дуг. Дуги не можуть бути проведені з вузла до вузла, або з переходу до переходу, а тільки з вузла до переходу і навпаки. Вузли з, яких виходять дуги до переходів називаються вхідними вузлами. Вузли, в які входять дуги з переходів називаються вихідними. Існує три типи переходів: миттєвий, випадковий та фіксований. Дослідники також запропонували політику управління дорожнім рухом, що направлена на забезпечення пріоритетного проїзду екстреного транспорту, шляхом управління сигналами світлофорів.

П. Б. Мірчандані та ін. [24] розробили стратегію для систем пріоритету екстреного транспорту, що базується на аналізі транспортного потоку, контролі сигналів світлофорів і швидкій адаптації до змін. Автори назвали стратегію повторною оптимізацією на перехрестях на основі категорій прибуття (Categorized Arrivals – based Re-optimization at Intersections, CAPRI). CARPI може бути використана для розробки різноманітних систем пріоритету, наприклад залізничних, екстреного транспорту та ін. Головна перевага використання CAPRI при розробці таких систем це те, що вона працює з будь-якими типами механізмів аналізу трафіка, такими як сенсори тиску, індуктивні петлі тощо. CAPRI використовує динамічне програмування для швидкої адаптації до змін в реальному часі. Як тільки стає відоме поточне місцезнаходження автомобіля швидкої допомоги і його напрямок, система на основі CAPRI може в реальному часі прораховувати найбільш оптимальний шлях. При цьому головний фокус стратегії CAPRI – забезпечити якомога менший вплив на весь дорожній рух, забезпечивши якомога швидший проїзд швидкої допомоги.

А. Луаті та ін. [25] запропонували систему пріоритету сигналів дорожнього руху (Traffic Signal Priority System, TSPS), що використовує мульти-агентні системи (MAS), Алгоритм Longest Queue First - Maximal Weight Matching (LQF-MWM) та стратегію пріоритету.

Запропонована система складається з агентів, кожен з яких розміщується на перехресті. Агент – базова частина системи, кожен з агентів слідує певному набору правил. Розроблений авторами алгоритм LQF-MWM аналізує кількість транспортних засобів на кожній смузі і планує послідовність сигналів світлофора

так, щоб пріоритет був у смуги, на якій найбільша черга транспортних засобів (Longest Queue First, LQF). І так до тих пір, поки всі смуги не проїдуть. Потім цей процес починається з початку. Якщо система помічає екстрений транспортний засіб, агент, що його помітив спочатку збирає інформацію про автомобіль а потім посилає запит іншим агентам, що знаходяться за суміжних перехрестях на отримання інформації про ситуацію на дорозі. LQF-MWM алгоритм аналізує інформацію отриману від сусідніх вузлів та інформацію про автомобіль швидкої допомоги. По перше – він перевіряє місце призначення швидкої допомоги, потім дивиться, яке з сусідніх перехресть знаходиться на оптимальному шляху до місця призначення. Агент, що знаходиться на наступному перехресті сповіщається про наближення автомобіля швидкої допомоги включаючи інформацію про смугу, по якій він їде. Як тільки сповіщення відіслане, система контролю світлофорами робить смугу, по якій їде екстрений транспортний засіб найбільш пріоритетною. Таким чином автомобілі на цій смузі зможуть проїхати першими не дивлячись на ситуацію на інших смугах. Використовуючи інформацію про швидкість, поточне розташування, тип та розміри автомобіля тощо, сигнали світлофора можуть бути заплановані так, щоб вплив на дорожній рух на перехресті був мінімальний.

Автори пропонують декілька способів виявлення екстрених транспортних засобів таких як Радіо-частотна ідентифікація (RFID), інфрачервоні сенсори прямої видимості тощо.

М. Абдус та ін. [26] запропонували використовувати Q-навчання для контролю транспортних потоків на перехрестях. Автори зазначають, що система повинна складатися з чотирьох інтелектуальних агентів, кожен з яких розміщується на відповідній дорозі і відслідковує кількість автомобілів.

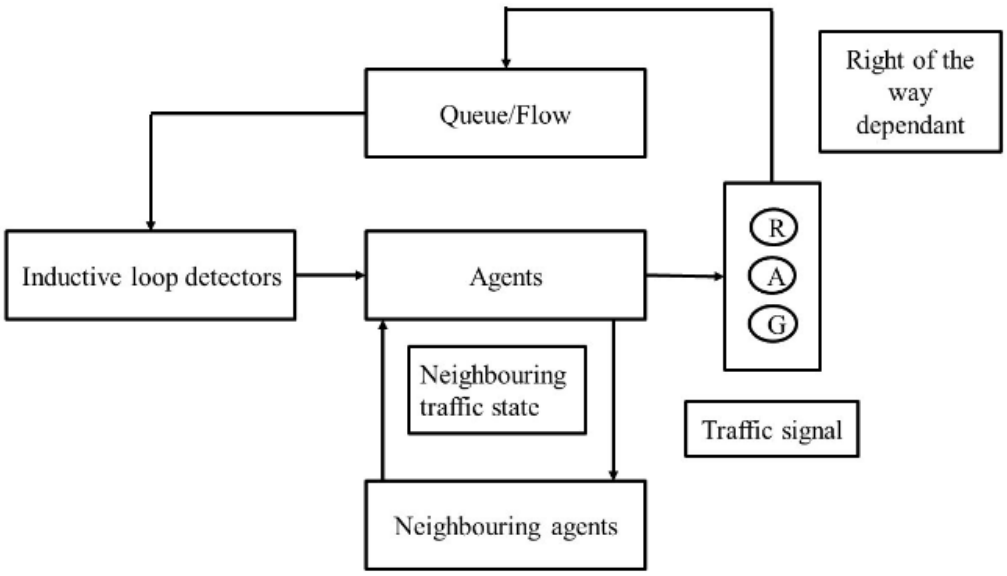


Рисунок 1.12 – Діаграма системи Б. П. Гокулан та ін.

Б. П. Гокулан та ін. [27] запропонували використовувати GFMAS для контролю руху на різних типах перехресть. За словами авторів перевага цього методу в тому, що можуть бути враховані і оброблені як заплановані так і не заплановані сценарії дорожнього руху. Діаграма системи запропонованої авторами представлена на рис. 1.12.

Таблиця 1.1 – порівняльна характеристка систем пріоритету екстрених транспортних засобів

Посилання	Тип	Технологія	Опти-мальний маршрут	Керу-вання трафіком
1	2	3	4	5
В. Тран та ін. [4]	Акустика	Ансамблева нейронна мережа	Ні	Так
Елліс Д. П. та ін. [5]	Акустика	Синусоїдальне моделювання	Ні	Так

Продовження таблиці 1.1

1	2	3	4	5
Бертеллі та ін. [6]	Акустика	Нейронна мережа	Ні	Так
У. Е. Брілл та ін. [10]	Акустика	Нейронна мережа	Ні	Так
Меуччі та ін. [11]	Акустика	MDF	Ні	Так
С. Шибуя та ін. [12]	Прямої видимості	Інфрачерворні датчики	Так	Так
Р. Бхарадвадж та ін. [13]	Прямої видимості	RFID	Ні	Так
А. Мазум та ін. [14]	Прямої видимості	Ультразвукові датчики	Ні	Так
В. Кодайр та ін. [15]	GPS	GPS, Xbee	Ні	Так
Е. С. Елтаіб та ін. [16]	GPS	GPS, GSM	Ні	Так
А. Бухеншайт та ін. [17]	Без-дротовий зв’язок	VANET	Ні	Так
Р. Сундар та ін. [21]	Без-дротовий зв’язок	Zigbee	Ні	Так
М. Масуд та ін. [22]	Вузли	WSN, CTP	Ні	Так
Ю. С. Хуанг [23]	Вузли	Мережі Петрі	Ні	Так
П. Б. Мірчандані та ін. [24]	Вузли	CAPRI	Так	Так
А. Луаті та ін. [25]	Вузли	LQF-MWM	Так	Так
М. Абдус та ін. [26]	Вузли	Q-Навчання	Ні	Так
Б. П. Гокулан та ін. [27]	Вузли	GFMAS	Ні	Так

В даному підрозділі ми розглянули існуючі варінти систем організації пріоритету екстрених транспортних засобів. В таблиці 1 наведено порівняльну характеристику розглянутих систем в залежності від того чи розраховують вони оптимальний маршрут та чи керують дорожнім рухом. Система вважається такою, що керує дорожнім рухом, якщо вона змінює в реальному часі схему руху на перехрестях та/або надає інформацію водіям про наближення екстреного транспорту. Кожна розглянута вище система в тій чи іншій мірі керує дорожнім рухом, але тільки декілька з них прораховують оптимальний маршрут.

1.2 Аналіз методів розпізнавання звуку

Як було визначено раніше, система розпізнавання передбачає в своїй основі виділення примітиву. Відповідно, для системи розпізнавання звуку, необхідно зрозуміти, що таке звук, і як відрізнити один елемент від іншого.

З точки зору фізики, звук це хвиля. Зустрічаючись з об’єктом, звукова хвиля відбивається від нього, тим самим створюючи зміни в повітряному середовищі. Людина розрізняє звук, коли зміни досягають вуха і починають впливати на барабанну перетинку. Далі мозок вирішує завдання розпізнавання образів. Кожна звукова хвиля має низку характеристик:

- форма;
- частота;
- амплітуда;
- фаза.

Базовою формою хвилі вважається – синусоїда. Саме вона визначає інші характеристики. Однак в реальності, як правило, хвиля, це не одна синусоїда, а поєднання декількох, що продемонстровано на рис. 1.13.

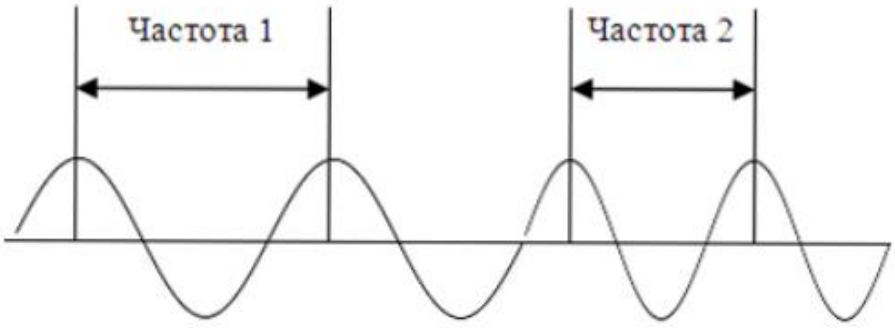


Рисунок 1.13 –Частота звукової хвилі.

Частота розраховує кількість скоєних коливань в 1 секунду. Амплітуда звукової хвилі безпосередньо пов'язана з гучністю звуку, а саме, чим більше амплітуда, тим голосніше звук. Якщо дві звукові хвилі мають однакову частоту і амплітуду, це означає, що вони знаходяться в фазі. Зміна фази походить від 0 до 360. Тут 0 означає повне поєднання хвиль, а 360 - протифазу.

Знаючи амплітуду і частоту звукової хвилі далі можливо перетворити її в набір математичних значень, щоб в подальшому проводити маніпуляції саме з ними. Тут слід позначити основні проблеми, які зустрічаються при роботі зі звуковими файлами:

1. Можлива присутність додаткових звукових хвиль. Таке може відбуватися через природню наявність додаткових звуків (наприклад, виділення співу конкретного птаха із загального фону лісових шумів) або через різні звукові ефекти (наприклад, наявності відлуння), тощо;
2. Один і той же звук у різних, навіть подібних об'єктів, може мати відмінності. різна амплітуда призведе до того, що один з звуків буде голосніше, другий - тихіше. Інтенсивність частоти коливання звукової хвилі вплине на те, чи буде звук низьким або високим, що характерно, наприклад, для людської мови.
3. Один і той же звук може видаватися повільніше або швидше. Наприклад, коли людині треба сказати щось дуже швидко, або ж мова ведеться розмірено.

4. Якість апаратної частини, яка приймає і передає звуковий сигнал на подальшу обробку.
5. Якщо на одному записі відбувається накладення двох сигналів, амплітуда яких не збігається, то це призводить до повного зникнення звуку.

Узагальнений алгоритм системи розпізнавання звуку можна записати в такий спосіб:

1. Отримання звукового сигналу;
2. Усунення сторонніх звукових сигналів і посилення корисного;
3. Ідентифікація корисного сигналу;
4. Оцінка якості розпізнавання.

За принципом побудови весь масив алгоритмів ділиться на три великі групи, проте всі вони передбачають у своїй основі еталонну базу, з елементами якої порівнюється сигнал. Причому порівняння може проводитися двома способами: або шляхом накладення спектрограми сигналу, або шляхом порівняння вже оцифрованих значень.

При роботі зі спектрограмами використовується метод динамічного тимчасового деформування (Dynamic Time Warping - DTW), при якому стартовий сигнал ділиться на кілька частин. Важливою особливістю тут є те, що кожен наступний відрізок починається не з того моменту, де закінчився попередній, а на деяку кількість фаз раніше, тобто містить в собі закінчення попереднього відрізка. далі частоти порівнюються за допомогою перетворення Фур'є.

Дійсно, нехай є дві звукові послідовності:

$$R = r_1, r_2, r_3, \dots r_n$$

$$\text{та } P = p_1, p, p_3, \dots p_m,$$

тоді для них можна розрахувати (формула 1.1):

$$DTW(R,P) = min\left\{\frac{\sum_{k=1}^{K} d(w_k)}{K}\right\}, \quad (1.1)$$

де K – коефіцієнт для нормалізації тимчасових послідовностей різної тривалості, $d(w_k)$ – матриця відстаней, що розраховується за формулою 1.2:

$$d(w_k) = (r_i - c_j)^2 \tag{1.2}$$

Складність таких систем оцінюється, як O(nm).

Іншим методом, який застосовується в системах розпізнавання звуку, є використання прихованих марковських моделей (Hidden Markov Model - HMM). Прихована марковська модель використовує систему кінцевих автоматів, яка складається з деякої кількості станів. Оскільки для даного алгоритму не важлива їх послідовність, така особливість отримала назву прихованих станів.

Звідси в методі робляться два обов'язкових допущення:

1. Звуковий сигнал розбивається на фрагменти, кожному з яких, відповідає певний стан в моделі, таким чином, що характеристики всередині одного фрагмента не змінюються.
2. Стан попередніх і наступних фрагментів не впливають на ймовірності один одного, вагу має тільки поточний стан.

Позначимо марковську модель - λ і визначимо для неї наступні параметри: нехай матриця переходів між станами позначається як (1.3):

$$A = \{a_{ij}\}, \tag{1.3}$$

де a_{ij} - ймовірність конкретного переходу між станами i і j.

Нехай загальна кількість станів в системі - N; M - значення, які приймає система в будь-якому з станів; Матрицю, що визначає ймовірність вихідних значень позначимо як (1.4):

$$B = \{b_j(k)\}, \tag{1.4}$$

де $b_j(k)$ - визначає для стану j ймовірність випадання параметра k.

Матрицю, що визначає ймовірність потрапляння в початковий стан позначимо як (1.5):

$$\Pi = \{\pi_i\}, \tag{1.5}$$

де π_i - ймовірність потрапляння системи в стан *i* в початковий момент часу.

Тоді прихована марковська модель визначається як (1.6):

$$\lambda = \{A, B, \Pi\} \tag{1.6}$$

Для того, щоб при практичному застосуванні система давала результати, які відповідають критеріям якості, необхідно вирішити також ряд підзадач, до яких віднесемо:

- Вибір алгоритму, що дозволяє нормалізувати використовувані вектора;
- Визначення параметрів N і M;
- Вибір алгоритму сегментації для початкового етапу роботи алгоритму;
- Еталонна база повинна бути велика і містити вичерпну кількість примітивів і їх поєднань.

Наступним методом роботи зі звуковими файлами є використання штучних нейронних мереж. Принцип дії системи показаний на рис. 1.14.

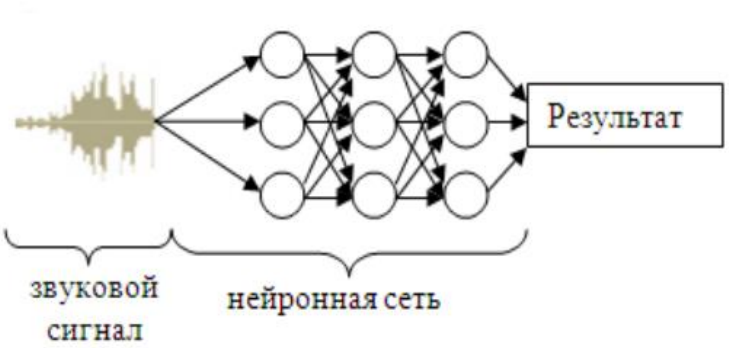


Рисунок 1.14 – Принцип розпізнавання звуку на основі нейронної мережі

На вхід нейронної мережі подається звуковий сигнал. Далі велика кількість вхідних параметрів піддаються обробці, аналогічній класифікації даних, після чого розпізнана інформація надходить на вихід системи відповідно до подальших завдань.

Варто зауважити, що в порівнянні з системами розпізнавання графічних образів, системи технічного слуху, засновані на нейронних мережах, ще відстають за якістю отриманого результату. Це обумовлено, в першу чергу недостатньо великої еталонною базою. Однак, з огляду на пристосованість нейронних мереж до самонавчання, а також враховуючи можливість побудови унікальної мережі для кожної конкретної задачі, це один з найбільш перспективних напрямків для задач розпізнавання звуків

1.3 Постановка задачі

Метою роботи є покращення контролю світлофорів на перехресті з метою забезпечення пріоритетного проїзду спеціального транспорту: автомобілів швидкої допомоги, поліцейських, пожежних, тощо. Для цього необхідно розробити надійну систему пріоритету екстрених транспортних засобів. Визначимо основні етапи, що необхідно виконати для виконання задачі::

- Провести аналіз існуючих методів і засобів контролю світлофорів для проїзду спеціального транспорту.
- Запропонувати метод розпізнавання спеціальних транспортних засобів на перехрестях.
- На основі даних аналізу спроектувати систему пріоритетного проїзду екстрених транспортних засобів. Розробити проектну документацію, визначити основні складові частини та компоненти.
- На основі проекту розробити систему для розпізнавання наявності спеціального транспорту та контролю світлофорів на перехресті.
- Провести експериментальні випробування розробленої системи.

1.4 Висновки

В першому підрозділі даного розділу було детально розглянуто та проаналізовано 27 робіт, що так чи інакше стосуються теми систем екстрених транспортних засобів. Було визначено основні методи та підходи до створення таких систем. Було проаналізовано сильні і слабкі сторони кожної з них, та зроблено відповідні висновки.

У другому підрозділі було розглянуто природу звукових хвиль, загальні підходи та методи в задачах розпізнавання звуку та основні проблеми, які можуть виникати в процесі розробки системи розпізнавання звуку. Також було розглянуто та проаналізовано три основних підходи до реалізації розпізнавання звуку а саме:

- Метод динамічного тимчасового деформування
- Приховані марковські моделі
- Нейронні мережі

В останньому підрозділі було виконано постановку задачі. Було визначено основні завдання та етапи даної роботи.

Вдало поєднуючи існуючі підходи із новими розробками та покращеннями можна створити систему, що буде здатна значно зменшити час проїзду екстрених транспортних засобів через перехрестя, що в свою чергу дозволить автомобілям швидкої допомоги, поліцейським та пожежникам швидше діставатися до місця призначення.

2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ ДОДАТКУ

2.1 Аналіз і обґрунтування вибору мови програмування

Одним з ключових моментів розробки програмного продукту є вибір правильної мови програмування. Обрана мова повинна бути якомога краще пристосована для задоволення усіх вимог і потреб, тобто мати весь потрібний набір бібліотек, інструментів та засобів які будуть потрібні для розробки програмного забезпечення. Неправильний вибір мови програмування може значно сповільнити та ускладнити процес розробки. Тому остаточний вибір повинен бути максимально зваженим та обґрунтованим. Було проведено порівняльний аналіз чотирьох популярних мов програмування а саме:

1. C++
2. C#
3. Java
4. Python

Мова C++ була розроблена Бьорном Страуструпом в підрозділі Bell Labs компанії AT&T у якості доповнення до мови C. C++ додав безліч нових можливостей у мову C. Його популярність була викликана об'єктно-орієнтованістю мови. Зараз C++ широко використовується для розробки програмного забезпечення, будучи однією з найпопулярніших мов програмування. З його допомогою створюють операційні системи, різноманітні прикладні програми, драйвери пристроїв, ігри та ін.

Серед найпопулярнішого програмного забезпечення, написаного на C++ (або з його використанням), знаходяться СУБД MySQL, інтернет-браузер Mozilla Firefox, більшість програмного забезпечення від Microsoft: операційні системи сімейства Windows, IDE Visual Studio, Internet Explorer, Microsoft Office. Adobe Photoshop, Adobe Illustrator та Adobe Premiere Pro повністю написані на C++. Також ця мова лежить в основі ігрового рушія Unity3D.

Java — це досить універсальна, об'єктно-орієнтована мова програмування, що часто використовується для веб-розробки та розробки під Android. Java була

розроблена компанією Sun Microsystems (надалі придбаною компанією Oracle). Дата офіційного випуску – 23 травня 1995 року.

Програми Java транслюються в байт-код, який потім виконується віртуальною машиною Java (JVM). JVM – це програма, яка обробляє байтовий код та передає інструкції обладнанню як інтерпретатор. Перевагою подібної реалізації є незалежність байт-коду від операційної системи та обладнання, що дозволяє виконувати Java-програми на будь-якому пристрої, для якого існує JVM.

Іншою важливою особливістю технології Java є гнучка система безпеки завдяки тому, що виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

Часто до недоліків концепції віртуальної машини відносять те, що виконання байт-коду віртуальною машиною може знижувати продуктивність програм та алгоритмів. Програми, написані на Java, мають репутацію повільніших і займаючих більше оперативної пам'яті, ніж написані мовою Сі. Однак, якщо порівнювати Java з інтерпретованими мовами, продуктивність Java зазвичай помітно вище.

C# - мова програмування, що поєднує об'єктно-орієнтовані та контекстно-орієнтовані концепції. Ця мова була розроблено в 1998-2001 роках групою інженерів в компанії Microsoft, як основна мова розробки додатків для платформи Microsoft .NET. Компілятор з C# входить до стандартної комплектації самої .NET, тому програми на ньому можна створювати і компілювати навіть без спеціальних інструментальних засобів на кшталт IDE Visual Studio.

C# відноситься до сім'ї мов з C-подібним синтаксисом, її синтаксис найбільш близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, навантаження операторів, покажчики на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників - мов C++, Delphi, Modula і Smalltalk - C#,

спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так C# не підтримує множинне успадкування класів (на відміну від C++) або виведення типів (на відміну Haskell).

Python — це високорівнева мова програмування загального призначення, яка використовується навіть для розробки веб-додатків. Мова орієнтована на підвищення продуктивності розробника і якості коду.

Python підтримує кілька парадигм програмування: структурну, об'єктно-орієнтовану, функціональну, імперативну та аспектно-орієнтовану. У мові присутня динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень та зручні високорівневі структури даних. Програмний код на Python організовується у функції та класи, які можуть об'єднуватись у модулі, а вони у свою чергу можуть бути об'єднані у пакети. Python зазвичай використовується як інтерпретований, але може бути скомпільований в байт-код Java та в MSIL (в рамках платформи .NET).

В таблиці 2.1 наведено результати порівняння розглянутих мов програмування за обраними критеріями.

Таблиця 2.1 – Порівняння мов програмування

Назва критерію	C#	C++	Java	Python
Пристосованість для наукових розробок, розрахунків, моделювання тощо.	0	0	0	1
Бібліотеки для візуалізації даних	1	1	1	1
Статична типізація	1	1	1	0
Швидкість розробки	1	0	1	1
Простота та читабельність	0	0	0	1
Сумарний коефіцієнт	3	2	3	4

Згідно з проведеним аналізом, мова програмування Python має перевагу перед іншими мовами, тому для реалізації програмного додатку було обрано саме її, адже вона задовольняє всі необхідні вимоги для створення програмного додатку і має велику кількість бібліотек для наукових розрахунків, аналізу, візуалізації тощо.

2.2 Аналіз і обґрунтування вибору середовища розробки

Дуже важливим кроком для розробки програмного продукту є вибір інтегрованого середовища розробки. Інтегроване середовище розробки (IDE) – це програма, призначена для розробки програмного забезпечення.

PyCharm (рис. 2.1) - це інтегроване середовище розробки для Python, яке має повний комплект засобів, необхідних для ефективного програмування.

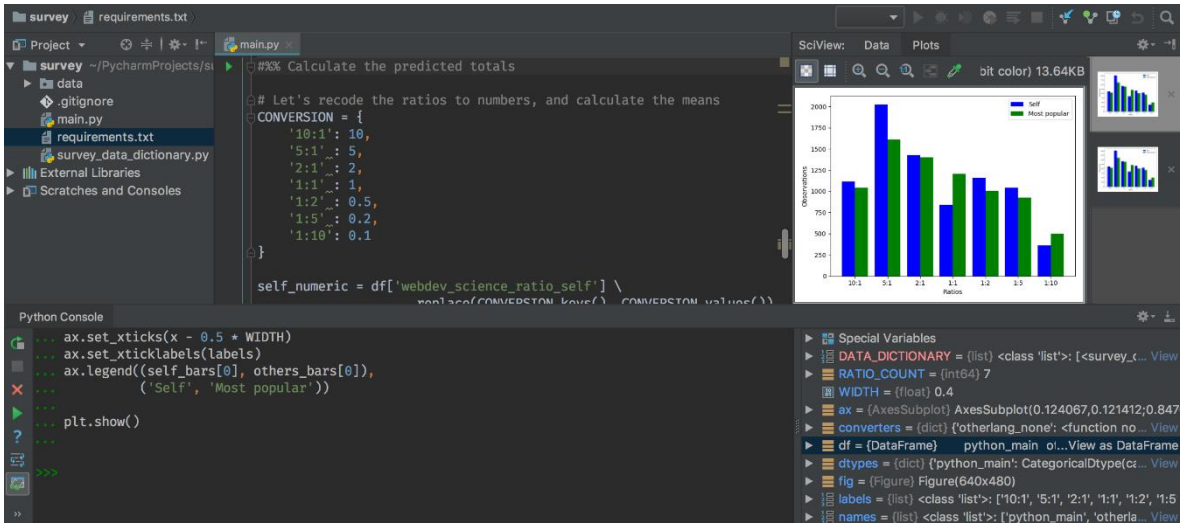


Рисунок 2.1 – Приклад інтерфейсу PyCharm

Зараз PyCharm поширюється в двох варіантах: платному (PyCharm Professional Edition) і безкоштовному (PyCharm Community Edition). Безкоштовна версія має відкритий вихідний код і поширюється під ліцензією Apache 2. Це полегшене середовище, яке підходить для розробки тільки на Python. Платний варіант являє собою більш розширену і функціональну версію з

можливістю розробки в тому числі багатомовних веб-додатків. Professional Edition підтримує фреймворки:

– Django,
– Flask,
– Google App Engine,
– Pyramid,
– web2py.

Spyder (рис. 2.2) - відкрите середовище розробки для Python. Воно є кроссплатформенним і доступним для Windows, Linux і MacOS. Назва Spyder розшифровується як Scientific PYthon Development EnviRonment, тобто наукове середовище розробки для Python. Програма створювалася для наукових розрахунків, і в цій сфері вона дійсно зручна.

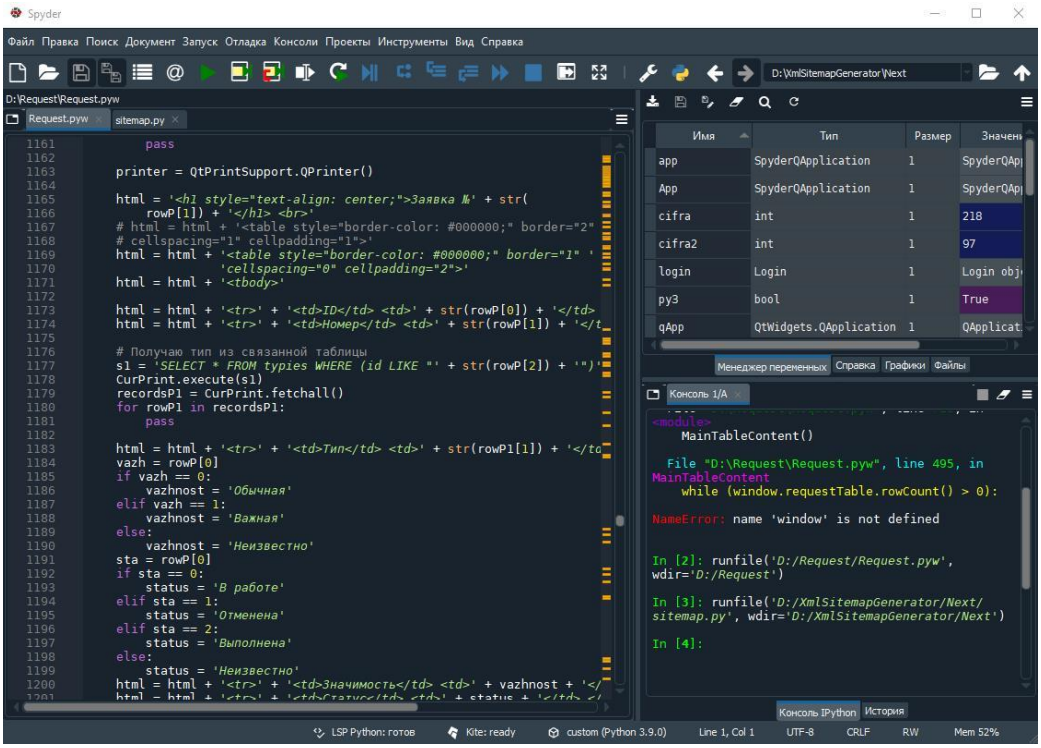


Рисунок 2.2 –Інтерфейс Spyder

Згідно зі словами розробників Spyder є:

1. Потужним інтерактивним середовищем розробки для мови програмування Python з просунутими можливостями редагування, інтерактивного тестування, налагодження та інтроспекції.
2. Середовищем чисельних розрахунків завдяки підтримці IPython (покращений інтерактивний інтерпретатор Python) і популярних бібліотек, таких як NumPy, SciPy (обробка даних, чисельні розрахунки та ін.) І matplotlib (інтерактивна 2D / 3D візуалізація).
3. Частиною модуля spyderlib для Python, який надає потужні віджети на PyQt4, такі як редактор коду, консоль Python (вбудована в додатку), редактор списків / кортежів і масивів NumPy.

Інтегроване середовище розробки Visual Studio - це стартовий майданчик для написання та налагодження коду, а також подальшої публікації додатків.

Visual Studio (рис 2.3) являє собою багатофункціональну програму, яку можна використовувати для різних аспектів розробки програмного забезпечення. Крім стандартного редактора і відладчика, які існують в більшості середовищ, Visual Studio включає в себе компілятори, засоби автозавершения коду, графічні конструктори і багато інших функцій для спрощення процесу розробки.

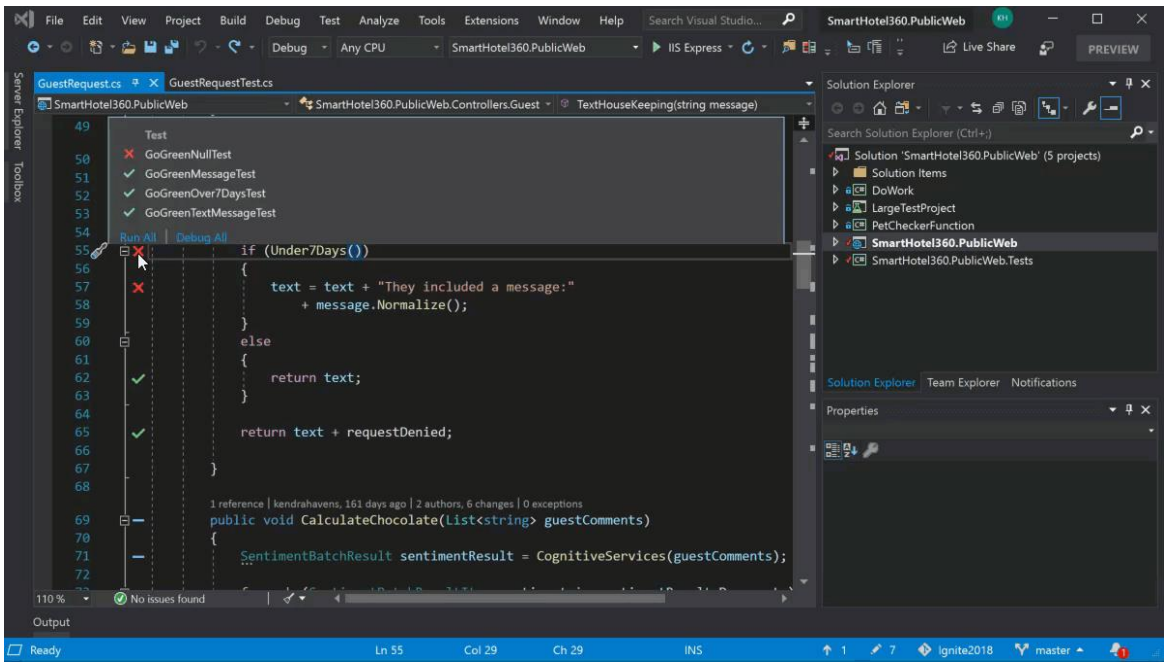


Рисунок 2.3 – Інтерфейс Visual Studio

Python Tools for Visual Studio (PTVS) дозволяє писати на Python в Visual Studio, включаючи наявність інструмента під назвою Intellisense, що надає зручний режим налагодження, розумного рефакторінгу та ін.

В таблиці 2.2 наведено результати порівняння IDE для розробки на мові програмування Python.

Таблиця 2.2 – Порівняння засобів розробки

Назва критерію	Microsoft Visual Studio	PyCharm	Spyder
Крос-платформенність	0	1	1
Простота у налагоджені	0	1	1
Автодоповнення	1	1	1
Система контролю версій	1	1	0
Функціональність	1	1	0
Сумарний коефіцієнт	3	5	3

В результаті проведення аналізу засобів розробки програмного додатку сумарний коефіцієнт PyCharm – 5, що переважає коефіцієнти інших аналогів, тому дане середовище програмування є оптимальним для даної розробки та вирішення поставленої задачі.

2.3 Аналіз засобів для реалізації програмного додатку

Для роботи було вирішено використовувати дистрибутив мов програмування Python і R під назвою Anaconda. Він включає набір популярних відкритих бібліотек, об’єднаних проблематикою науки про дані та машинного навчання. Основна ціль — поставка єдиним узгодженим комплектом найбільш використовуваних відповідними колами користувачів тематичних модулів (таких, як NumPy, SciPy, Astropy та ін.) з вирішенням виникаючих залежностей і

конфліктів, яких неможливо уникнути, якщо встановлювати кожен компонент окремо.

Основна особливість дистрибутива — оригінальний менеджер пакетів conda з графічним інтерфейсом Anaconda Navigator, що дозволяє відмовитися від стандартних менеджерів пакетів (таких, як pip для Python). Дистрибутив завантажується один раз і вся наступна конфігурація, в тому числі встановлення додаткових модулів, може проводитися оффлайн. Крім того, забезпечується можливість створення кількох ізольованих середовищ з роздільною обробкою залежностей.

Підтримуються платформи Linux (x86-64), Windows (i686, x86-64), macOS. Розповсюджується за ліцензією BSD, існує також комерційна версія (Anaconda Enterprise).

Anaconda Navigator - це графічний інтерфейс (GUI), включений у дистрибутив Anaconda (рис. 2.4), що дозволяє запускати додатки, встановлювати додаткові пакети і т.д. без використання командного рядка.

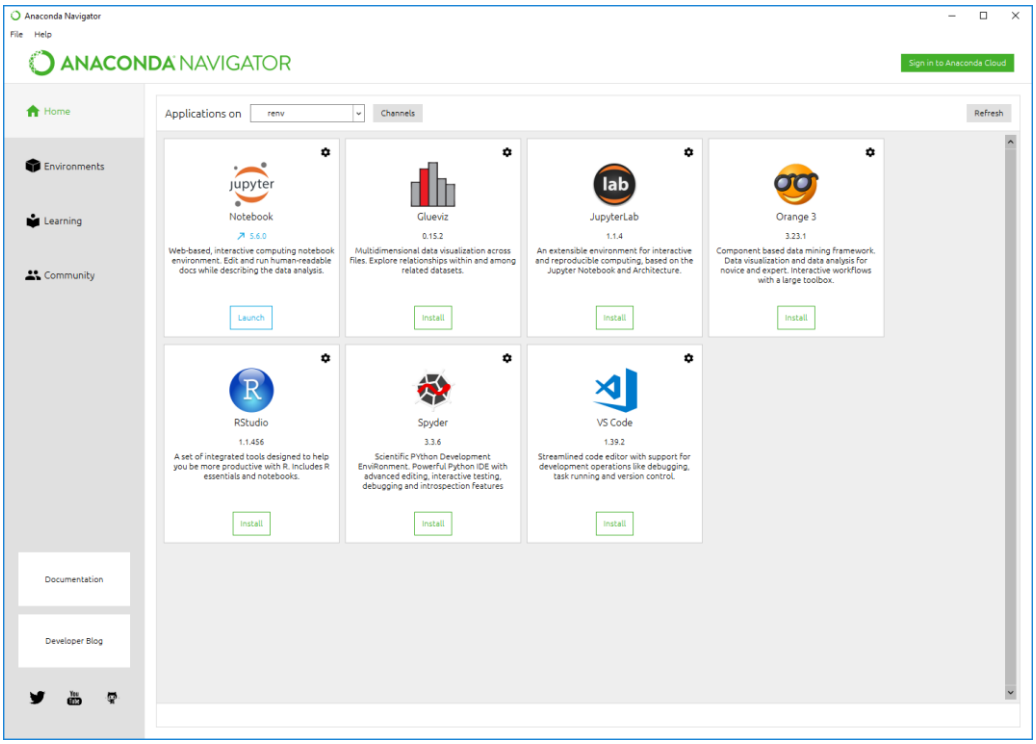


Рисунок 2.4 – Графічний інтерфейс програми Anaconda Navigator

За замовчуванням в Anaconda Navigator доступні такі додатки:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange
- RStudio
- Visual Studio Code
- PyCharm CE

Розглянемо основні бібліотеки, що використовувалися в роботі.

Для роботи зі звуком використовувалася бібліотека librosa. Librosa — це пакет Python для аналізу музики та аудіо. Він надає інструменти, необхідні для отримання інформації зі звукових файлів, її обробки, аналізу та візуалізації.

1. Librosa.load - завантажує аудіо-файл у якості послідовності значень амплітуди. В процесі завантаження дані автоматично переводяться під задану частоту дискретизації.

Основні параметри:

- path (string) – шлях до файлу;
- sr (int) – частота дискретизації (за замовчуванням 22050);
- mono (bool) – перетворити у моно-сигнал (за замовчуванням True).

2. Librosa.stft – віконне перетворення Фур’є. Переводить сигнал з часового простору у частотний шляхом ділення даних на маленькі частини, що перекриваються і обчислення для кожної дискретного перетворення Фур’є (DFT).

Основні параметри:

- y (array) – вхідні дані;
- win_length (int) – розмір вікна обчислення.

3. Librosa.amplitude_to_db та Librosa.db_to_amplitude – перетворення сигналу з амплітудної розмірності в децибельну та навпаки.

4. Librosa.display.specshow – відображення спектрограми, хромограми тощо.

Основні параметри:

– data (2d-array) – двомірні дані для відображення;
– sr (int) – частота дискретизації (за замовчуванням 22050).

Для реалізації фільтру Баттерворта було використано бібліотеку SciPy. SciPy - це бібліотека Python з відкритим вихідним кодом, призначена для вирішення наукових та математичних проблем. Вона побудована на базі NumPy та дозволяє керувати даними, а також візуалізувати їх за допомогою різних високорівневих команд.

1. SciPy.butter – створює фільтр Баттерворта n-го порядку та повертає його коефіцієнти.

Основні параметри:

– n (int) – значення порядку;
– wn (array) – частота або частоти зрізу;
– btype (string) – тип фільтру (lowpass, highpass, bandpass, bandstop).

2. SciPy.lfilter – фільтрує вхідний сигнал, використовуючи цифровий фільтр.

Основні параметри:

– b (array) – коефіцієнти чисельника;
– a (array) – коефіцієнти знаменника;
– x (array) – вхідні дані.

Для роботи з радіопередавачем було використано бібліотеку rpi-rf. Це модуль Python для надсилання та отримання 433/315 МГц LPD/SRD сигналів недорогими радіо-модулями, що через GPIO підключаються до Raspberry Pi (рис 2.5).

Рисунок 2.5 – Типовий радіо-модуль для Raspberry Pi

2.4 Висновки

Розглянуто та проведено аналіз переваг і недоліків 4-ох популярних мов програмування. В результаті було визначено мову Python. Це пов’язано з тим, що вкрай важливо якомога швидше отримати робочий прототип розробки, навіть якщо для цього буде потрібно поступитися у ефективності та швидкості роботи. На початкових етапах роботи необхідно мати можливість швидко змінювати функціонал, переписувати одні модулі і видаляти інші. Швидкість адаптації до зовнішніх умов, що постійно змінюються, багато в чому визначає те, чи буде проект успішним. Саме тому був обраний Python – оскільки це мова, філософія якої полягає у простоті, швидкості розробки динамічності та адаптивності.

Розглянуто основні середовище розробки для мови Python. Було проаналізовано їх переваги та недоліки. Враховувалися функціональні можливості, зручність інтерфейсу, простота у використанні, наявність додаткових можливостей тощо. В результаті було обрано середовище розробки PyCharm, як таке, що найбільш відповідає потребам і вимогам на даний момент часу.

Проведено аналіз основних інструментальних засобів, що використовувалися для розробки. Було розглянуто Python-дистрибутив Anaconda та його графічний інтерфейс Anaconda-Navigator. Крім цього було наведено основні бібліотеки, що використовувалися в ході роботи та опис їх методів.

3. ОПИС МЕТОДІВ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис принципу роботи системи

Незважаючи на те, що дана система може використовуватися на будь яких типах перехресть, для простоти будемо розглядати типове перехрестя на 4 дороги. Схема такого перехрестя наведена на рис. 3.1.

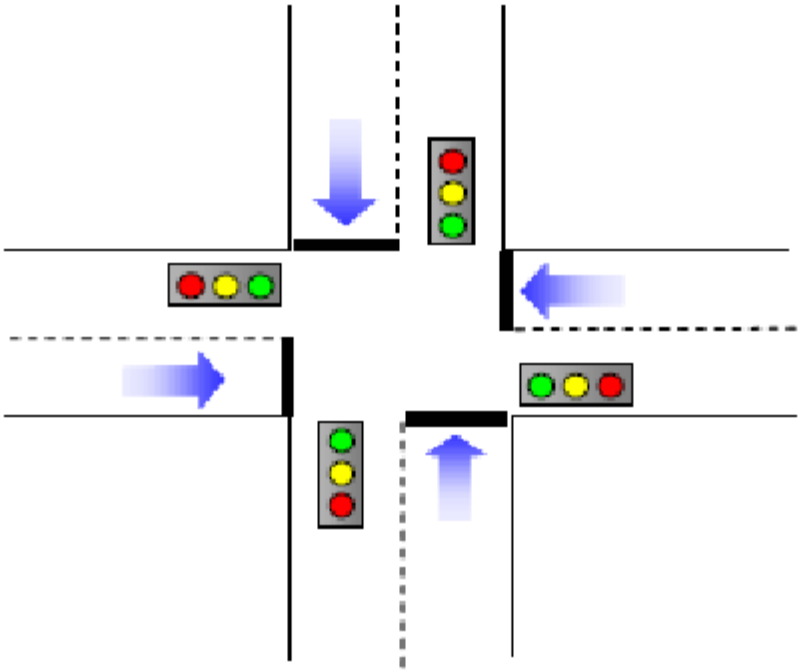


Рисунок 3.1 – Схема типового перехрестя на 4 дороги

Робота будь якої системи пріоритету екстреного транспорту базується на декількох основних кроках:

1. Розпізнавання автомобіля спеціального призначення серед інших автомобілів.
2. Вибір оптимальної стратегії для забезпечення проїзду такого автомобіля через перехрестя.
3. Виконання обраної стратегії.

В залежності від реалізації у кожному конкретному випадку на цей каркас може додаватися додатковий функціонал, такий як, наприклад, розрахунок

найкращого маршруту, повідомлення водіям про наближення спеціального транспортного засобу тощо.

За способом розпізнавання такі системи можна поділити на дві великі групи:

– Системи, що базуються на отриманні сигналів від автомобіля. Вони відрізняються більшою надійністю але і значно більшою складністю та вартістю впровадження, оскільки потребують встановлення на кожний автомобіль датчиків, сенсорів, пристроїв для бездротового зв’язку тощо.
– Системи, що базуються на алгоритмах розпізнавання спеціальних транспортних засобів на основі звукових сигналів та зображення з камер. Надійність такої системи напряму залежить від надійності роботи алгоритму машинного навчання, що використовується для розпізнавання. В той же час такі системи не потребують оснащення автомобілів додатковими пристроями.

В даній роботі було вирішено використовувати комбінацію розпізнавання звуку та розпізнавання зображення. Це значно зменшує вартість впровадження системи у промислових масштабах.

Комбінована система забезпечує значно більшу надійність, ніж система, що використовує тільки один спосіб розпізнавання.

Для розпізнавання сигналів на перехресті буде встановлюватися 4 апаратних вузли, кожен з яких оснащений камерою, мікрофоном та радіо-передавачем для зв’язку з іншими вузлами та базовою станцією.

Базова станція – центральний процесор системи, що приймає сигнали від кожного з чотирьох апаратних вузлів, обробляє та відправляє назад команди для перемикання світлофори. Для базової станції знадобиться досить потужний процесор і великий об’єм оперативної пам’яті, оскільки вона повинна постійно обробляти та аналізувати великі об’єми звукової та візуальної інформації. Крім цього вона також повинна бути обладнана радіо-передатчиком для зв’язку з вузлами. На рис. 3.2 зображено можливі варіанти обладнання світлофору у двох можливих варіантах.

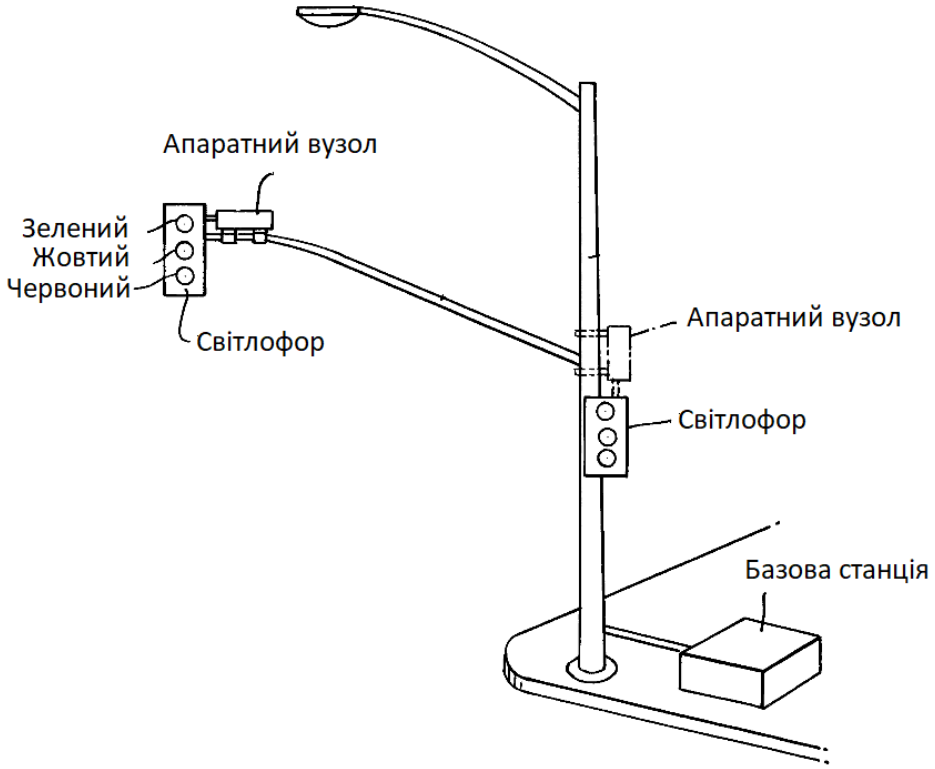


Рисунок 3.2 – Схема обладнання світлофору у двох можливих варіантах

Розглянемо загальний алгоритм роботи системи:

1. Перехрестя працює у штатному режимі. Звукові датчики, що розташовані у кожному апаратному вузлі, надсилають аудіо дані в базову станцію. Базова станція в свою чергу в реальному часі обробляє ці дані і приймає рішення про те, чи наявні в них звуки сирени.
2. Якщо базова станція розпізнає звук сирени, вона посилає всім апаратним вузлам сигнал на увімкнення камер. Аналіз аудіо-даних припиняється, базова станція починає аналізувати зображення.
3. Якщо алгоритм розпізнає вогні швидкої допомоги, поліцейського або пожежного транспорту, базова станція відправляє відповідному апаратному вузлу команду увімкнути зелений сигнал світлофору, а усім іншим – червоний сигнал.
4. Якщо на протязі деякого часу не було розпізнано вогні спеціального транспорту, алгоритм повертається до пункту 1.

На рис. 3.3 ображено діаграму роботи вищеописаного алгоритму. На рис. 3.4 зображено загальний принцип роботи системи.

Рисунок 3.3 – Діаграма алгоритму роботи системи

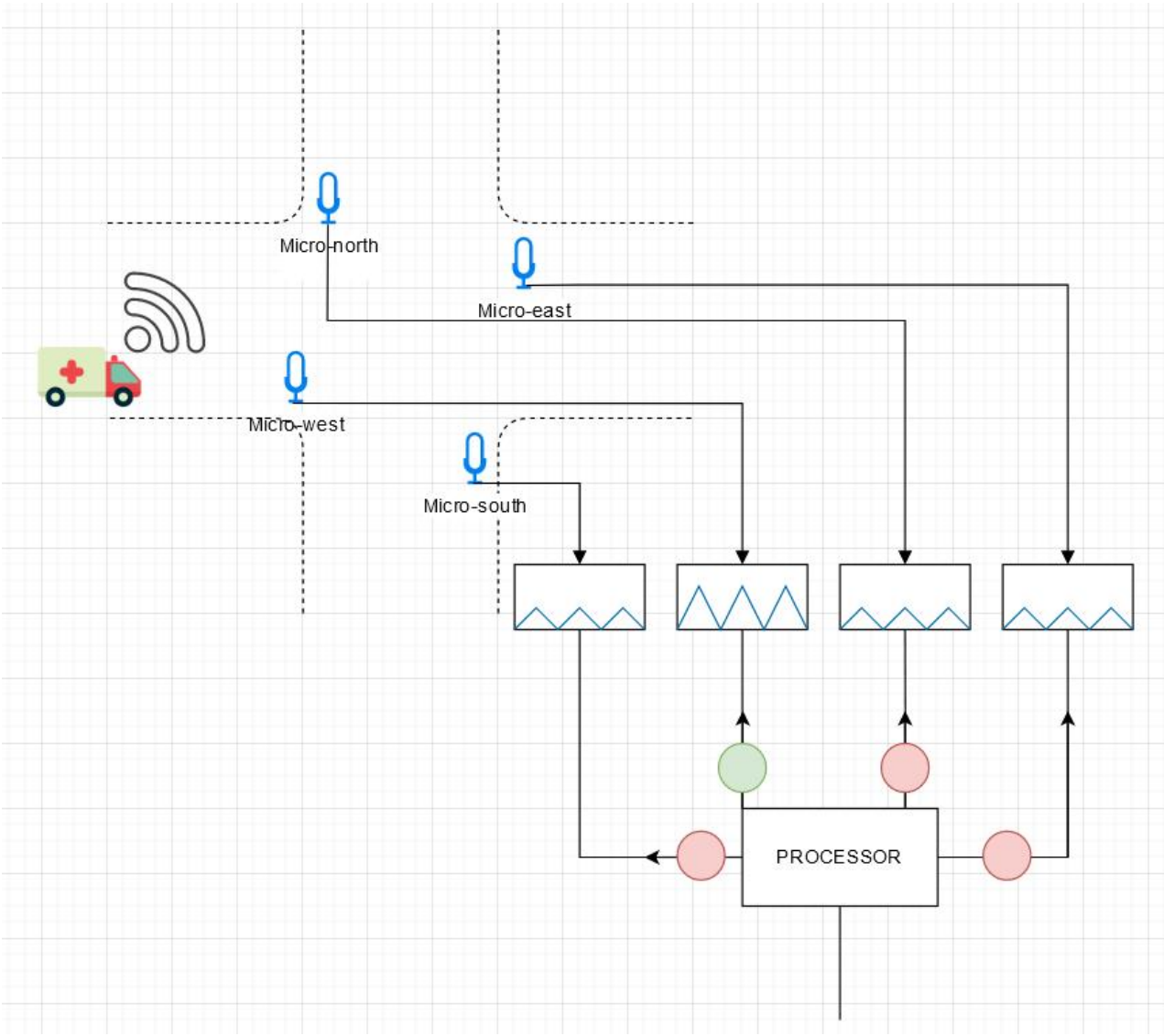


Рисунок 3.4 – Загальна схема роботи системи

3.2 Опис методу розпізнавання звуку

Як було визначено раніше, система розпізнавання передбачає в своїй основі виділення примітиву. Відповідно, для системи розпізнавання звуку, необхідно зрозуміти, що таке звук, і як відрізнити один елемент від іншого.

З точки зору фізики, звук це хвиля. Зустрічаючись з об'єктом, звукова хвиля відбивається від нього, тим самим створюючи зміни в повітряному середовищі. Людина розрізняє звук, коли зміни досягають вуха і починають впливати на барабанну перетинку. В мікрофоні роль барабанної перетинки виконує мембрана. Мікрофон знімає коливання повітря і перетворює їх у електричний сигнал. На рис 3.5 зображено приклад сигналу, який ми можемо отримати з мікрофону.

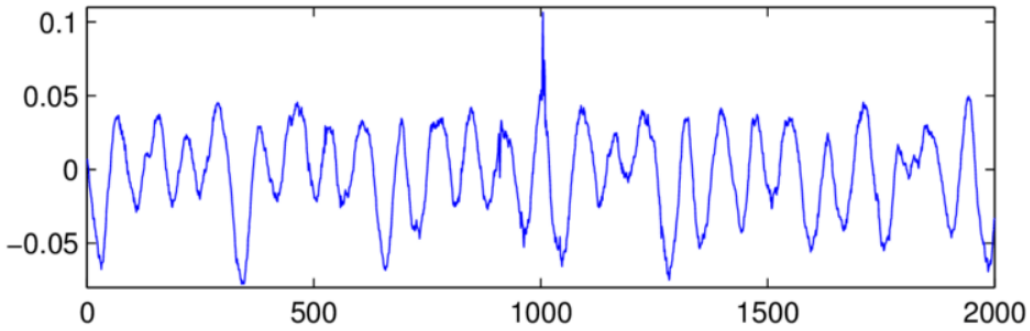


Рисунок 3.5 – Приклад звукового сигналу

Базовою формою звукової хвилі, як і будь-якої іншої є синусоїда, але в реальності звуковий сигнал як правило представляє собою поєднання декількох синусоїд з різними частотами (рис. 3.6).

Частота звукового сигналу розраховує кількість скоєних коливань в 1 секунду. Амплітуда звукової хвилі безпосередньо пов'язана з гучністю звуку, а саме, чим більше амплітуда, тим голосніше звук. Якщо дві звукові хвилі мають однакову частоту і амплітуду, це означає, що вони знаходяться в фазі. Зміна фази проходить від 0 до 360. Тут 0 означає повне поєднання хвиль, а 360 – протифазу.

Знаючи амплітуду і частоту звукової хвилі далі можливо перетворити її в набір математичних значень, щоб в подальшому проводити маніпуляції саме з ними.

Для того, щоб розпізнати будь-який звук, нам необхідно буде порівнювати отриманий аудіо-сигнал с деяким еталонним записом. Але проводити порівняння сигналів у часовому просторі не є досить зручним, оскільки сигнали можуть знаходитись у протифазі. Також на порівняння можуть впливати шуми. Тому є доцільним перевести сигнал з часового простору у частотний.

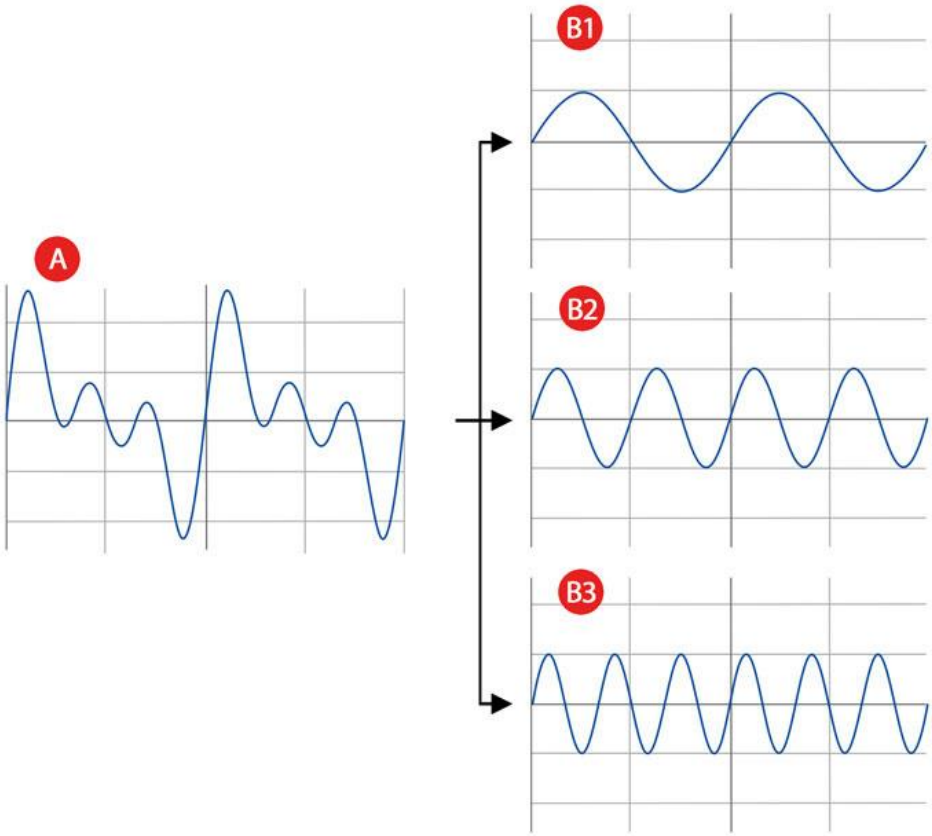


Рисунок 3.6 – Звукова хвиля найчастіше представляє собою поєднання декількох синусоїдальних хвиль

Важливо зазначити, що будь-який сигнал довільної форми можна представити у вигляді набору гармонічних сигналів різних частот. Іншими словами, сигнал складної форми в часовому просторі має набір комплексних відліків у частотній області, які називаються гармоніками. Ці відліки виражають

амплітуду та фазу гармонічного впливу на певній частоті. Чим більший набір гармонік у частотній області, тим точніше представляється сигнал складної форми.

Для перетворення сигналів у частотний простір використовується так зване перетворення Фур'є. Це математична операція, що дозволяє представити будь-яку функцію у вигляді набору гармонічних сигналів. Перетворення Фур'є є основою методів згортки і проектування цифрових кореляторів, активно застосовується при спектральному аналізі, використовується під час роботи з довгими числами.

Для прикладу проаналізуємо сигнал, що складається з трьох синусоїдальних хвиль з частотою 5, 10 та 20 Гц. відповідно (рис. 3.7).

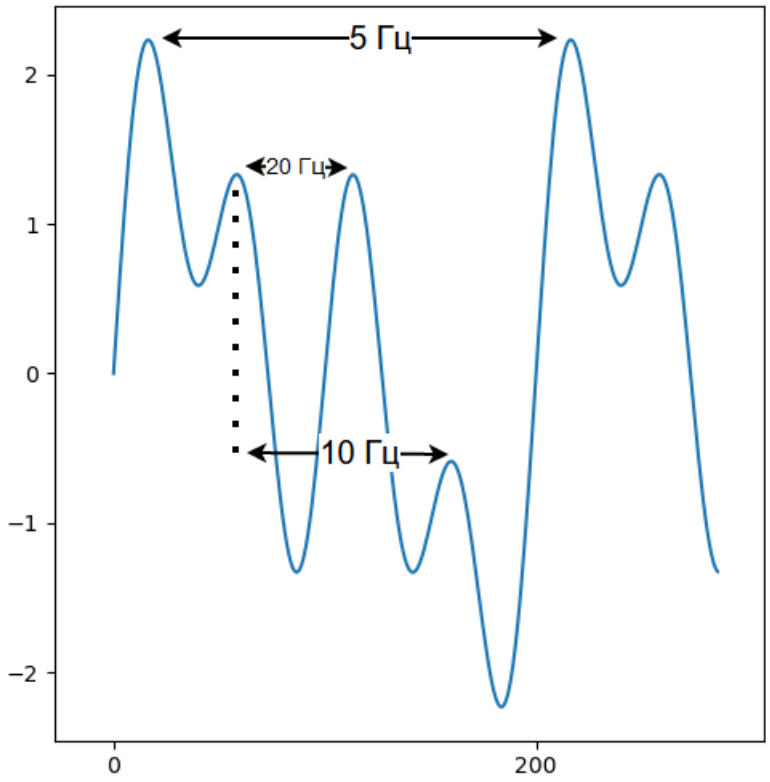


Рисунок 3.7 – Звуковий сигнал, що складається з частот 5, 10 та 20 Гц.

Формула такого сигналу буде виглядати наступним чином (3.1):

$$f(t) = \sin(10\pi t) + \sin(20\pi t) + \sin(40\pi t) \quad (3.1)$$

Застосуємо до наведеного сигналу формулу перетворення Фур'є (3.2):

$$\hat{f}(\omega) = \int_0^1 f(t) e^{-2\pi i \omega t} dt \quad (3.2)$$

Після перетворення отримуємо сигнал в частотному спектрі, тобто залежність амплітуди від частоти. На рис. 3.8 бачимо, що піки графіка знаходяться саме в частотах початкового сигналу: 5, 10 та 20 Гц. Таким чином, можемо працювати вже з окремими частотами, наприклад: виділяти важливі частоти, застосовувати частотні фільтри, зменшувати шум тощо.

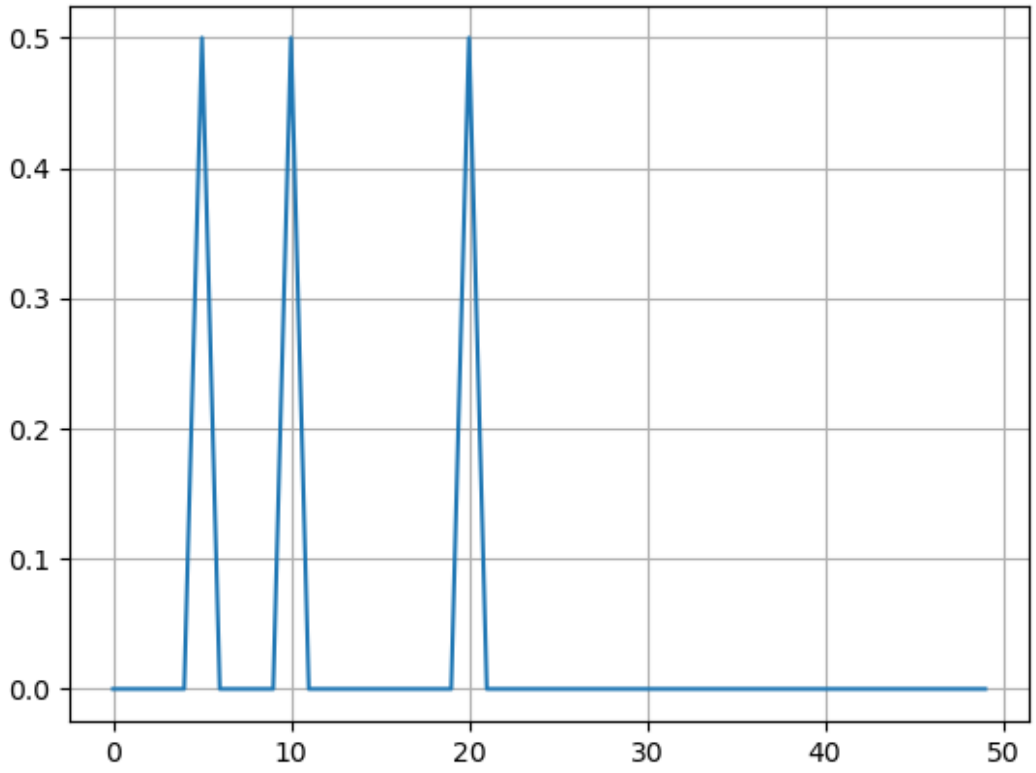


Рисунок 3.8 – Графік функції $\hat{f}(\omega)$ – звуковий сигнал в частотному просторі

На дорогах рідко буває зовсім тихо. Найчастіше звуки вулиці – це шум автомобілів, розмови людей, музика з навколишніх магазинів та ін. Тому

алгоритм для розпізнавання сирен повинен вміти знаходити потрібні йому звуки серед інших.

Звук сирени, як правило, знаходиться в невеликому частотному діапазоні. Таким чином, проаналізувавши еталонний запис, ми отримаємо дані про те, які частоти найбільш важливі для того, чи іншого типу сирени. Знаючи цю інформацію, ми можемо застосувати один або декілька відповідних частотних фільтрів до записаного сигналу, щоб виділити тільки потрібні частоти, відкинувши всі інші.

Існує три основних види фільтрів:

– Фільтр низьких частот – пропускає лише частоти, що нижчі за визначений поріг.
– Фільтр високих частот – пропускає лише частоти, що вищі за визначений поріг.
– Пропускаючий фільтр діапазону частот – пропускає лише частоти, що знаходяться у визначеному діапазоні.
– Затримуючий фільтр діапазону частот – пропускає всі частоти, що не знаходяться у визначеному діапазоні.

На рис. 3.9 зображено всі основні види фільтрів та їх комбінації.

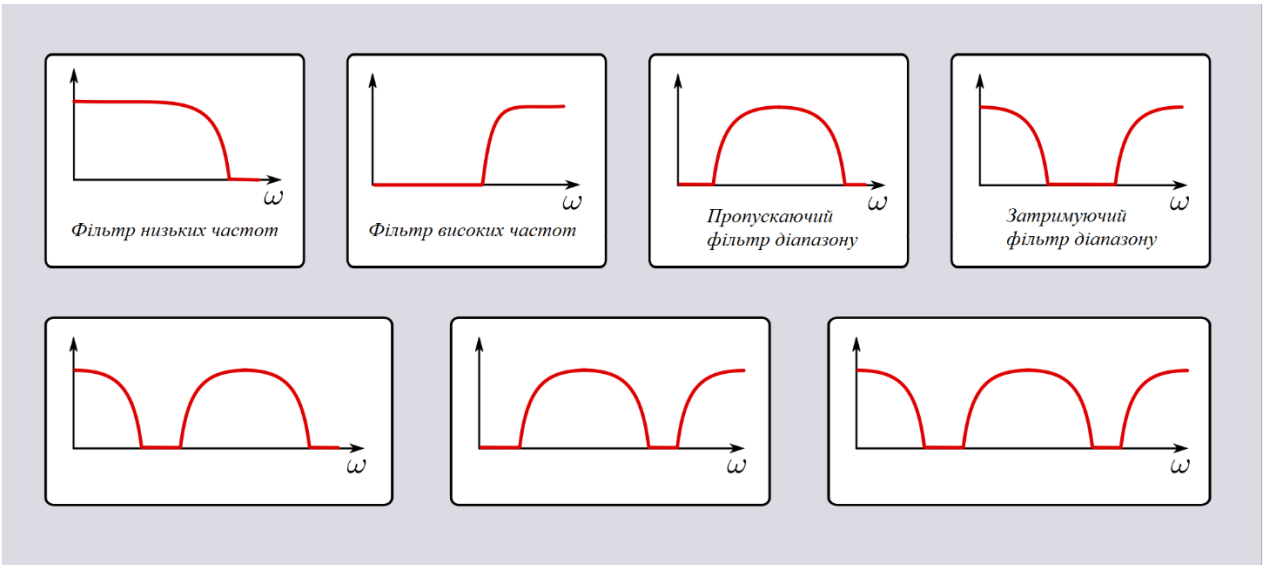


Рисунок 3.9 – Основні види фільтрів та їх комбінації

У даній роботі будемо використовувати пропускаючий фільтр діапазону частот, оскільки нам необхідно виділити з загальної звукової картини перехрестя, звук сирени, що знаходиться у вузькому частотному діапазоні.

Для фільтрації будемо використовувати фільтр Баттерворта. Фільтр Баттерворта – один з типів електронних фільтрів. Фільтри цього класу відрізняються від інших методом проектування. Фільтр Баттерворта проектується так, щоб його амплітудно-частотна характеристика була максимально гладкою на частотах смуги пропускання. На рис 3.10 зображено частотний відклик фільтрів Баттерворта 3-го, 6-го та 9-го порядків.

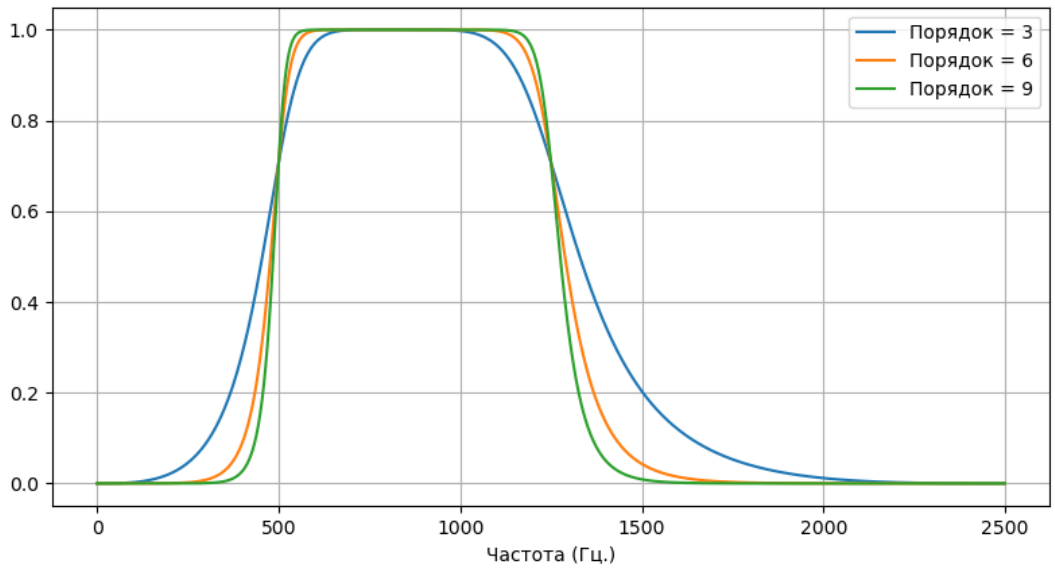


Рисунок 3.10 – Частотний відклик фільтрів Баттерворта 3-го, 6-го та 9-го порядків

Подібні фільтри були вперше описані британським інженером Стефаном Баттервортом у статті «Про теорію фільтруючих підсилювачів» (англ. On the Theory of Filter Amplifiers) [29], у журналі Wireless Engineer у 1930 році.

Враховуючи вищесказане, можна сформувати послідовність дій для визначення наявності звуку сирени в записаному аудіо-сигналі:

1. Необхідно проаналізувати еталонний сигнал, знайти в ньому діапазон або декілька діапазонів, в яких знаходяться найважливіші частоти.
2. До записаного аудіо-сигналу застосувати перетворення Фур'є, тобто перенести його з часового простору в частотний.
3. До даних, отриманих на другому кроці, застосувати частотні фільтри, відповідні до діапазонів, визначених на першому кроці.
4. Порівняти сумарну амплітуду відфільтрованого сигналу з заданим пороговим значенням і прийняти рішення про наявність звуку сирени у сигналі.

3.3 Проектування інформаційної системи

Інформаційна система складається з двох основних частин: програмне забезпечення базової станції та програмне забезпечення апаратного вузла, що розміщується на світлофорі. На рис. 3.11 зображено концептуальну схему інформаційної системи.

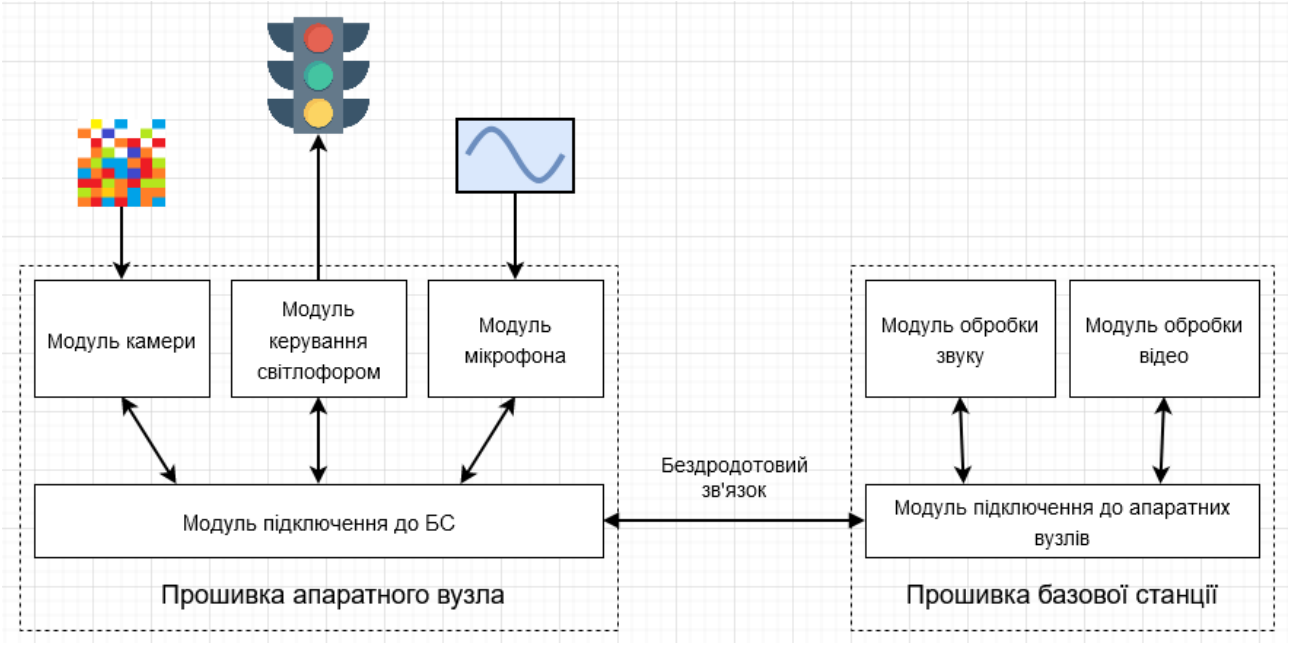


Рисунок 3.11 – Концептуальна схема інформаційної системи

Для апаратного вузла необхідно реалізувати наступний функціонал:

1. Модуль керування світлофором.

2. Модуль, що відповідає за запис звуку з мікрофона.
3. Модуль, що відповідає за запис відео з камери.
4. Модуль, що відповідає за попередню обробку та відправку інформації на базову станцію, а також за отримання та обробку команд від базової станції.

Відповідно для базової станції необхідні наступні функції:

1. Модуль, що відповідає за підключення до вузлів, прийом та обробку даних, а також за надсилання команд.
2. Модуль, що відповідає за обробку звукових даних та розпізнавання в них звуку сирени.
3. Модуль, що відповідає за обробку відео-даних та розпізнавання сигнальних вогнів.

На рис. 3.12 наведено діаграму послідовності алгоритму інформаційної системи:

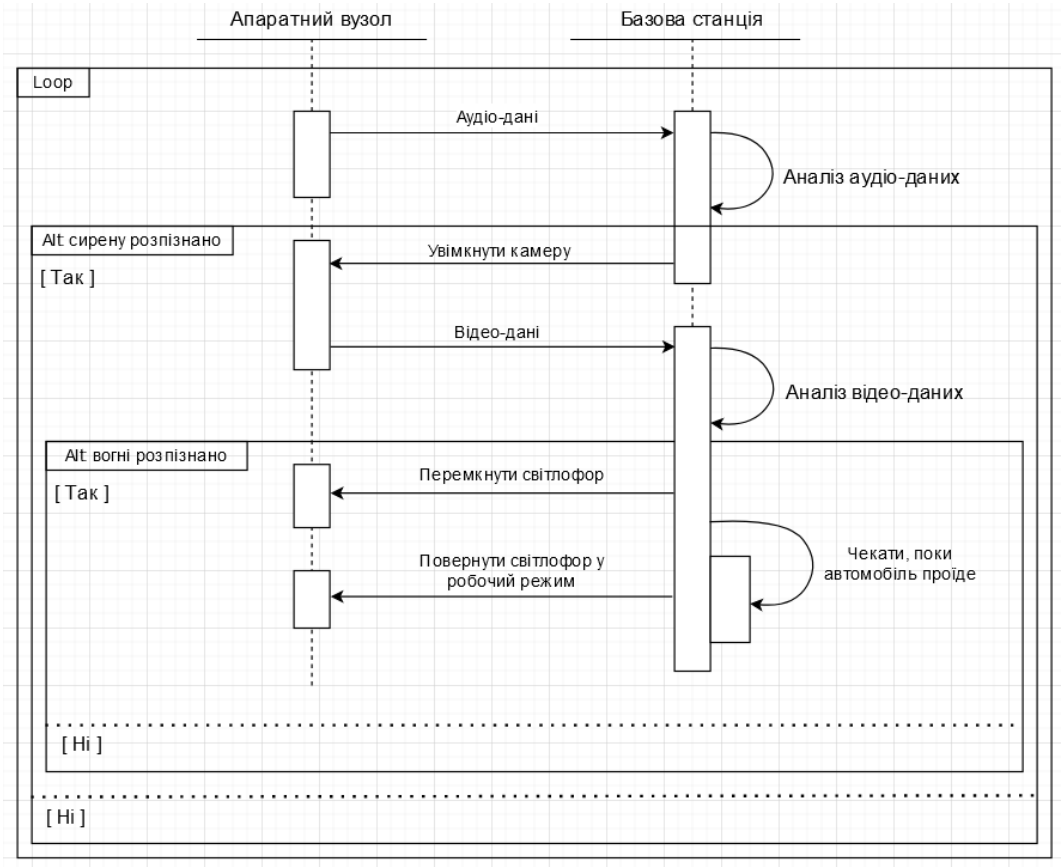


Рисунок 3.12 – Діаграма послідовності алгоритму інформаційної системи

3.4 Програмна реалізація інформаційної системи

Перший крок до розпізнавання звуку сирени – це аналіз існуючого еталонного запису. Для цього необхідно виконати наступні кроки:

1. Завантажити аудіо-дані з файлу.
2. Перенести аудіо-дані з часового простору у частотний.
3. Візуалізувати отримані дані.
4. Вручну вибрати найбільш важливі частоти для даного типу сирени.

   Для прикладу будемо використовувати чистий звук сирени автомобіля швидкої допомоги, взятий з відкритих джерел. За допомогою перетворення Фур’є (FFT, fast Fourier transform) переведемо його у частотний простір та відобразимо результат на графіку (див. додаток Й).

На рис. 3.13 зображено аудіо-сигнал до та після перетворення Фур’є.

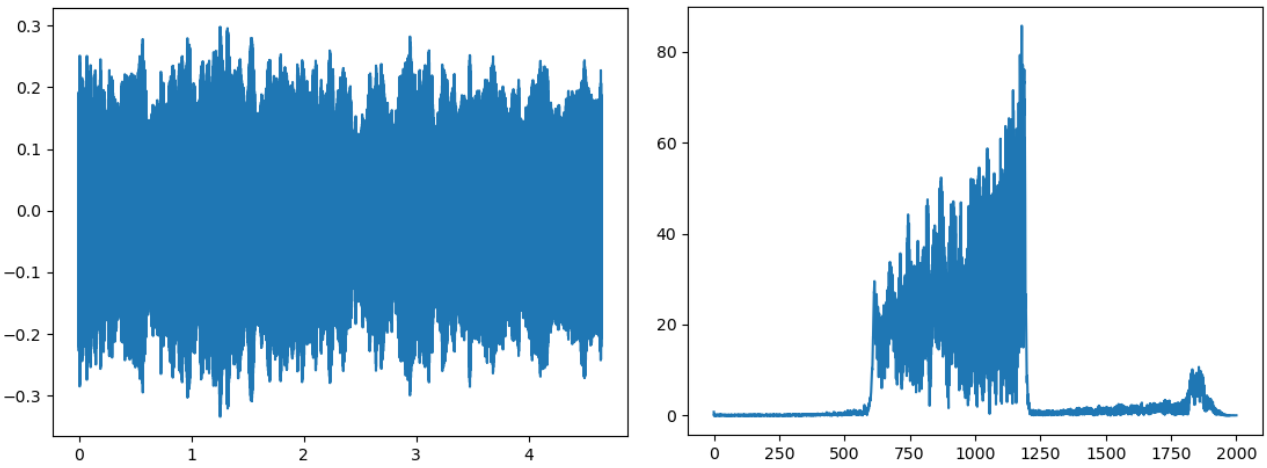


Рисунок 3.13 – Звук сирени у часовому (ліворуч) та частотному (праворуч) просторі

Для більшої наочності створимо також спектрограму. Отримаємо її за допомогою віконного перетворення Фур’є (STFT, short-time Fourier transform). Код, що відповідає за відображення спектрограми знаходиться у додатку Б. Віконне перетворення Фур'є — це перетворення Фур'є, що застосовується для визначення синусоїдної частоти та вмісту фази локальної секції сигналу, що має

властивість змінюватись в часі. Класичне перетворення Фур’є враховує спектр сигналу, який взято у всьому діапазоні існування змінної. Найчастіше інтереси зосереджуються тільки на локальному розподілі частот, у той час коли необхідно зберегти первинну змінну (зазвичай час). У цьому випадку використовується узагальнене перетворення Фур'є, так зване віконне перетворення Фур’є. На рис. 3.14 зображено отриману спектрограму.

Як ми бачимо основні частоти звуку зосереджені у діапазоні від 500 до 1250 Гц. Знаючи це, ми можемо відфільтрувати звук за допомогою фільтру Баттерворта. Порядок візьмемо рівним 1, оскільки нам необхідне найменше згладжування. На рис. 3.15 зображено спектрограми початкового та відфільтрованого звуків. Для наочності сигнал був переведений з амплітуди в децибели, оскільки так помітніше шум. Код, що відповідає за фільтрацію сигналу наведено в додатку В.

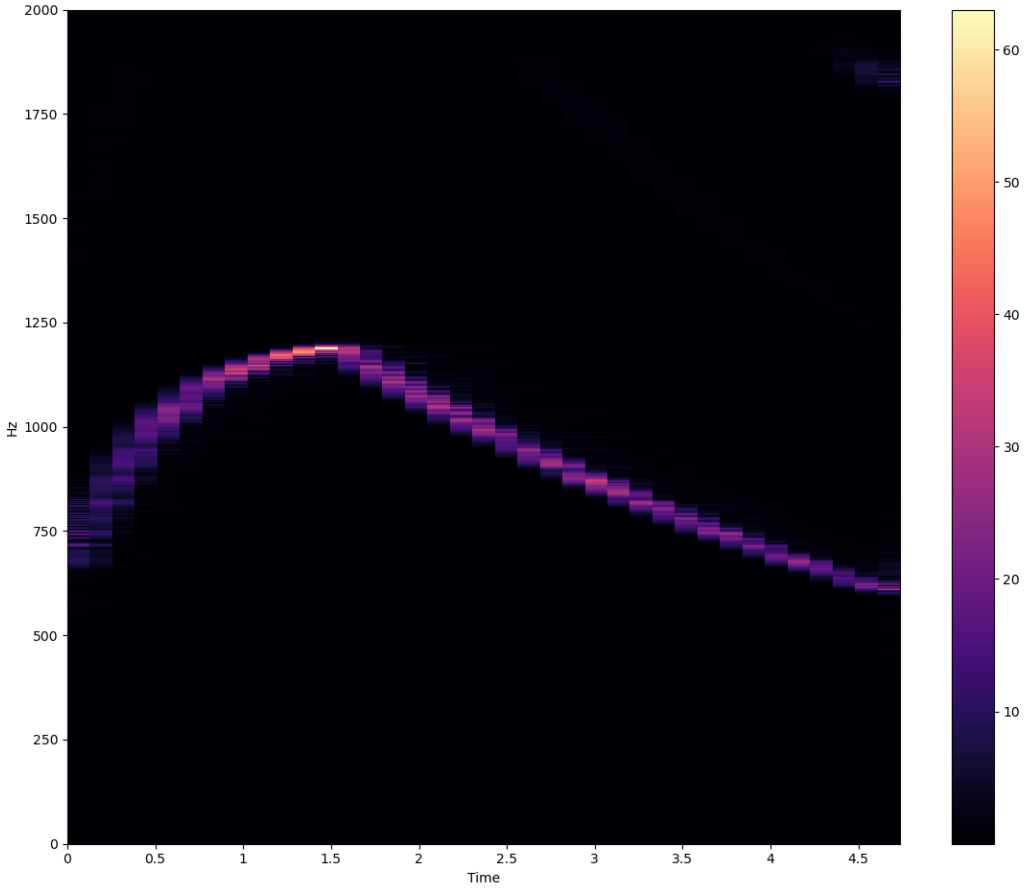


Рисунок 3.14 – Спектрограма еталонного звуку сирени

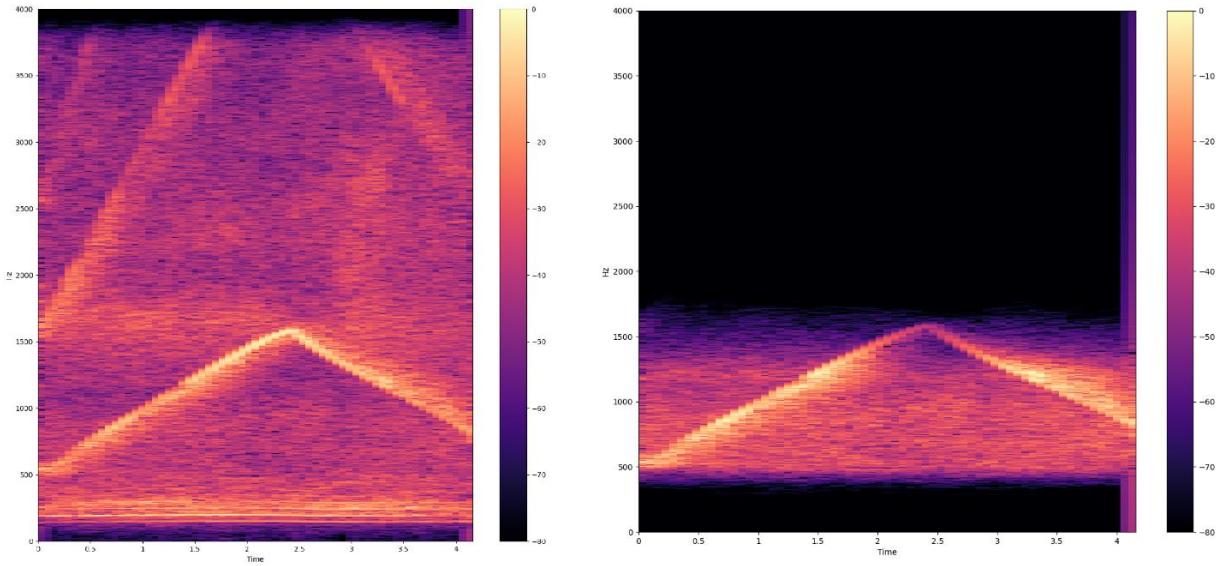


Рисунок 3.15 – Початковий (ліворуч) та відфільтрований (праворуч) сигнали у вигляді спектрограми.

Розглянемо алгоритм підключення апаратних вузлів до базової станції:

1. При ввімкненні апаратний вузол починає кожні 10 секунд надсилати запит на підключення, що включає в себе унікальний ідентифікатор вузла та ключ безпеки:

   {

   SenderId: 123e4567-e89b-12d3-a456-426614174000,

   Auth: xvz1evFS4wEEPTGEFPHBog:L8qq9PZyRg6ieKGE

   }

2. При отриманні такого запита, базова станція перевіряє ключ і якщо він правильний вносить ідентифікатор в свою базу та надсилає вузлу команду почати запис звуку:

   {

   ReceiverId: 123e4567-e89b-12d3-a456-426614174000,

   Auth: xvz1evFS4wEEPTGEFPHBog:L8qq9PZyRg6ieKGE,

   Command: 1 (StartRecording)

   }

3. Апаратний вузол починає записувати звук та надсилати ауді-дані невеликими пакетами на аудіо станцію:

   {

   SenderId: 123e4567-e89b-12d3-a456-426614174000,

   Auth: xvz1evFS4wEEPTGEFPHBog:L8qq9PZyRg6ieKGE

   Data: [300, 310, 354, 321, …]

   }

4. Раз в 10 секунд базова станція надсилає повідомлення для підтримки з’єднання, якщо апаратний вузол не отримує таке повідомлення, то зв'язок вважається розірваним і алгоритм повертається до кроку 1.

   Код, що відповідає за зв’язок з апаратними вузлами наведено у додатках Г та Д (надсилання повідомлень та прийом відповідно).

Розглянемо процес розпізнавання звуку сирени більш детально (див. додаток Ї). Кожен раз, коли модуль зв’язку отримує нову порцію аудіо-даних, він відправляє їх в центральний модуль, який в свою чергу зберігає дані в буфер (див. додаток Е). Коли загальна тривалість даних, що зберігаються в буфері стає більшою, ніж 2 секунди, запускається процес аналізу. Аудіо-дані проходять процес фільтрації, після чого амплітуда вихідного сигналу порівнюється з пороговим значенням. На основі цього приймається рішення про те, чи присутній звук сирени в аудіо-даних.

Розглянемо загальну структуру проекту та основні пакети (рис 3.16).

buffers - пакет, в якому зберігається код, що відповідає за буферізацію даних. На даний момент він представлений одним класом AudioBuffer (див. додаток Е). Пізніше, коли буде доданий функціонал для аналізу відео, буде створено інший клас – VideoBuffer.

connectors – пакет, що відповідає за з’єднання з апаратними вузлами. На даний момент представлений трьома класами. ConnectorBase – абстрактний клас, що задає інтерфейс для конкретних реалізацій. ConnectorRF – клас, що наслідує ConnectorBase та відповідає за зв’язок з апаратними вузлами через радіо-

передатчик. ConnectorMock – клас, що також наслідує ConnectorBase і використовується для тестів.

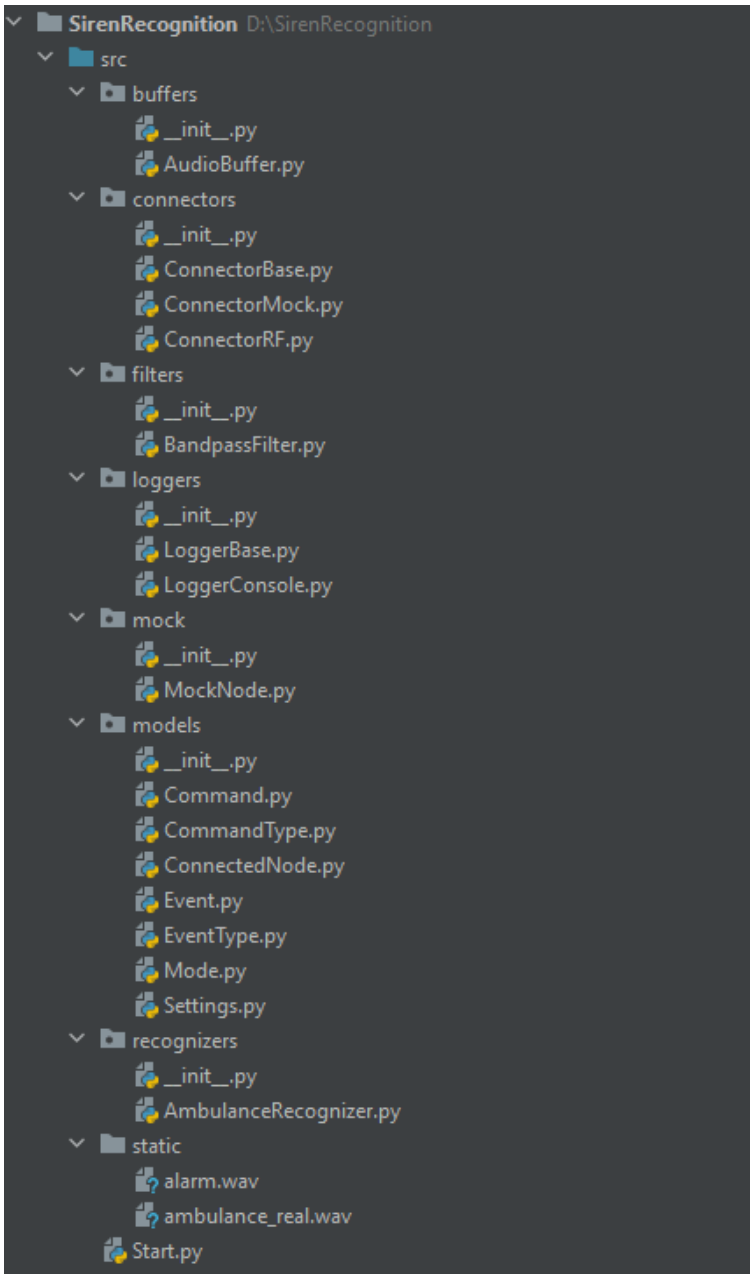


Рисунок 3.16 – Загальна структура проекту

filters – пакет, що представлений одним класом. BandpassFilter – реалізація фільтру Баттерворта.

loggers – тут знаходяться модулі, що відповідають за логування.

mock – представлений одним класом. MockNode – це програмна реалізація логіки апаратного вузла, що призначена для тестів. Цей клас імітує поведінку

справжнього апаратного вузла: надсилає повідомлення, читає звук з файлу і відправляє його частинами тощо.

models – моделі, що використовуються в програмі. Не містять бізнес-логіки.

recognizers – представлений одним класом. AmbulanceRecognizer – відповідає за розпізнавання звуку сирени (див. додаток Є).

start.py – файл, що представляє центральний модуль і відповідає за зв’язок всіх інших модулів між собою. Його повний код наведено у додатку Ж.

3.5 Висновки

Приведено базову схему роботи системи. Було вказано основні принципи дії та складові компоненти, а також базовий алгоритм роботи.

Детально розглянуто запропонований алгоритм розпізнавання звуку. Були наведені основні формули та розрахунки. Також було описано спосіб фільтрації. Було обґрунтовано вибір фільтра Баттерворта у якості смугового фільтра для використання в роботі.

Проведено концептуальне проектування системи. Було розроблено діаграму послідовності, що відображає кроки алгоритму. Також було створено діаграму, що показує основні модулі системи та зв’язки між ними.

Приведено опис програмної реалізації. Приведено фрагменти коду основних складових частин. Було розроблено та продемонстровано діаграму потоку даних, з якої видно основні програмні модулі та зв’язки між ними. Окрім цього було розглянуто загальну структуру проекту та її складові. Було детально описано всі пакети, з яких складається проект, їх складові та призначення.

4 ТЕСТУВАННЯ

4.1 Модульне-тестування системи

Модульне тестування (Unit Testing) – це тип тестування програмного забезпечення, при якому тестуються окремі модулі чи компоненти програмного забезпечення [32]. Його ціль полягає в тому, щоб перевірити, що кожна одиниця програмного коду працює належним чином. Цей вид тестування виконується розробниками на етапі кодування програми. Модульні тести ізолюють частину коду та перевіряють його працездатність. Одиницею для вимірювання може бути окрема функція, метод, процедура, модуль чи об'єкт.

У моделях розробки SDLC, STLC, V Model модульне тестування – це перший рівень тестування, який виконується перед інтеграційним тестуванням. Модульне тестування відноситься до класу методів тестування білої скриньки і зазвичай виконується розробником.

Відсутність модульного тестування при написанні коду значно збільшує рівень дефектів при подальшому (інтеграційному, системному та приймальному) тестуванні. Якісне модульне тестування на етапі розробки заощаджує час, а отже, зрештою, і гроші.

- Модульні тести дозволяють виправити помилки на ранніх етапах розробки та знизити витрати.
- Це допомагає розробникам краще розуміти кодову базу проекту та дозволяє їм швидше та простіше вносити зміни до продукту.
- Хороші юніт-тести є проектною документацією.
- Модульні тести допомагають із міграцією коду.

Модульне тестування поділяється на ручне та автоматизоване. І хоча програмна інженерія не виділяє перевагу одного над іншим, найчастіше використовується автоматизоване. При ручному модульному тестуванні зазвичай використовується покрокова інструкція. Алгоритм автоматизованого полягає в наступному:

- Розробник записує в програму одиницю коду, щоб протестувати її.

– Розробник може ізолювати одиницю коду більш якісного тестування. Ця практика передбачає копіювання коду у власне середовище тестування. Ізоляція коду допомагає виявити непотрібні залежності між кодом, що тестується, та іншими модулями.
– Для модульних тестів часто використовуються фреймворки. Використовуючи інфраструктуру автоматизації, розробник визначає критерії тесту для перевірки коректного виконання коду, і в процесі виконання тестових випадків реєструє невдалі. Багато фреймворків автоматично відзначають і повідомляють про невдалі тести і можуть зупинити подальше тестування, спираючись на серйозність збою.

Для модульного тестування в даній розробці використовувався фреймворк Python unittest. Це вбудований модуль, що підтримує автоматизацію тестів, використання загального коду для налаштування та завершення тестів, об'єднання тестів у групи, а також дозволяє відокремлювати тести від фреймворку для виведення інформації.

Для автоматизації тестів, unittest підтримує деякі важливі концепції:

– Тестовий стенд (test fixture) - виконується підготовка, необхідна для виконання тестів і всі необхідні дії для очищення після виконання тестів. Це може включати, наприклад, створення тимчасових баз даних або запуск серверного процесу.
– Тестовий випадок (test case) – мінімальний блок тестування. Він перевіряє роботу заданого модуля на різних наборах даних. Модуль unittest надає базовий клас TestCase, який можна використовувати для створення нових тестових випадків.
– Набір тестів (test suite) - кілька тестових випадків, наборів тестів або того й іншого. Він використовується для об'єднання тестів, які мають бути виконані разом.
– Виконавець тестів (test runner) - компонент, який керує виконанням тестів та надає користувачеві результат. Виконавець може використовувати

графічний або текстовий інтерфейс або повертати спеціальне значення, яке повідомляє результати виконання тестів.

Важливою концепцією в рамках розробки модульних тестів є Dependency injection (DI) або включення залежностей, що представляє механізм, який дозволяє зробити взаємодіючі в додатку об'єкти слабозв'язаними. Такі об'єкти пов'язані між собою через абстракції, наприклад, через інтерфейси, що робить всю систему гнучкішою, більш адаптивною і розширюваною [33].

Нерідко для встановлення залежностей у подібних системах використовуються спеціальні контейнери – IoC-контейнери (Inversion of Control). Такі контейнери є свого роду фабриками, які встановлюють залежності між абстракціями та конкретними об'єктами і, як правило, керують створенням цих об'єктів.

В даній роботі використовувався DI-фреймворк під назвою dependency-injector. Це надійне, добре задокументоване та підтримуване рішення для роботи з залежностями.

```
def test_append(self):
    #Arrange
    buffer = AudioBuffer(1000)

    #Act
    buffer.append(np.zeros(1000))
    data = buffer.getData()

    #Assert
    self.assertEqual(len(data), 1000)
```

Рисунок 4.1 – Приклад функції для тестування методу append класу AudioBuffer

На рисунку 4.1 зображено приклад функції, що тестує метод append з класу AudioBuffer. Розберемо детальніше, як це працює. Кожен тест-кейс складається з наступних частин:

– Arrange – налаштування середовища, створення необхідних залежностей. Ініціалізація контейнерів, сервісів тощо. В даному випадку лише створюється об’єкт буферу.
– Act – виконання певних дій, які тестуються. В даному випадку, ми додаємо у буфер масив з 1000 значень, після чого отримуємо данні, що зберігаються в буфері, використовуючи метод buffer.getData()
– Assert – перевірка правильності даних. У даному випадку ми перевіряємо чи збігається розмір масиву, що зберігається у буфері з тим, що в нього додали.

На рис. 4.2 зображено результат тестування 4-ох методів класу AudioBufferTests (див. додаток З).

```
FAILED (errors=1)
PS D:\SirenRecognition> cd src
PS D:\SirenRecognition\src> python -m unittest -v unittests/buffers/AudioBufferTests.py
test_append (unittests.buffers.AudioBufferTests.AudioBufferTests) ... ok
test_clear (unittests.buffers.AudioBufferTests.AudioBufferTests) ... ok
test_duration (unittests.buffers.AudioBufferTests.AudioBufferTests) ... ok
test_length (unittests.buffers.AudioBufferTests.AudioBufferTests) ... ok

----------------------------------------------------------------------
Ran 4 tests in 0.007s

OK
```

Рисунок 4.2 – Приклад консольного виводу тестування класу AuidoBuffer

4.2 Тестування алгоритму розпізнавання звуку

Тестування алгоритму розпізнавання звуку проводилося за методом білої скриньки. Тестування методом білої скриньки (також: прозорої, відкритої, скляної скриньки; засноване на коді або структурне тестування) – метод тестування програмного забезпечення, який передбачає, що внутрішня структура та реалізація системи відома тестувальнику. Ми вибираємо вхідні значення, ґрунтуючись на знанні коду, який їх оброблятиме. Так само ми знаємо, яким має бути результат цієї обробки. Знання всіх особливостей програми та її реалізації

– обов'язкові для цієї техніки. Тестування методом білої скриньки – поглиблення у внутрішнє влаштування системи, поза межі її зовнішніх інтерфейсів.

У таблиці 4.1 наведено переваги та недоліку даного методу тестування.

Таблиця 4.1 – Переваги та недоліки тестування методом білої скриньки

Переваги	Недоліки
Оскільки тестувальник знає вихідний код, стає дуже легко дізнатися, який тип даних може допомогти в ефективному тестуванні програми.	Для виконання тестування методом білої скриньки необхідна велика кількість спеціальних знань
Тестування може проводитися на ранніх етапах: немає необхідності чекати створення інтерфейсу користувача.	При використанні автоматизації тестування на цьому рівні, підтримка тестових скриптів може бути досить накладною, якщо програма часто змінюється.
Можна провести ретельніше тестування, з покриттям великої кількості шляхів виконання програми.	Важко підтримувати. Можуть знадобитися спеціалізовані інструменти, такі як аналізатори коду і інструменти. налагодження.

Для тестування було зроблено сім записів зі звуком сирени автомобіля швидкої допомоги в різних частинах міста, зокрема при різних погодних умовах, з різної відстані, для автомобіля, що находиться в русі або стоїть на місці тощо. Крім цього було записано інші звуки: шуми дорожнього руху, музику з придорожніх магазинів, звук потягу тощо.

Розглянемо тест-кейси, що були виконані (таблиця 4.2).

Таблиця 4.2 – Список тест-кейсів

№	Опис
1	2
1	Звук сирени швидкої допомоги у відносно тихому середовищі, записаний в ясну погоду. Окрім нього у записі присутні ледве помітний звук проїжджаючого повз автомобіля та звуки птахів. Автомобіль швидкої рухається на великій швидкості.
2	Звук сирени швидкої допомоги, записаний в центрі міста увечері, в ясну погоду. Автомобіль швидкої допомоги стоїть на світлофорі. У записі присутні помітні звуки інших автомобілів, музика з кафе, що знаходиться біля дороги та розмови людей.
3	Звук сирени швидкої допомоги, записаний в ясну погоду, вдень у передмісті. Автомобіль швидкої допомоги виїжджає із за повороту і, набираючи швидкість їде прямо по вулиці. Інші звуки майже відсутні.
4	Звук сирени швидкої допомоги, записаний під час сильного снігопаду в центрі міста. Автомобіль швидкої допомоги проїжджає перехрестя на великій швидкості. Окрім звуку сирени у записі присутні помітний звук розмов людей, звуки проїжджаючих автомобілів, гавкіт собаки.
5	Звук сирени швидкої допомоги, записаний під час сильного дощу на майже пустій вулиці. Автомобіль швидкої допомоги на великій швидкості проноситься повз. У записі присутні дуже помітні звуки дощу та ледве чутні розмови людей.
6	Звук сирени швидкої допомоги, записаний у дуже зашумленому середовищі в центрі міста, увечері, в ясну погоду. Інші звуки, що присутні у записі помітно впливають на загальну картину. Швидка стоїть на світлофорі потім починає рух та проїжджає повз.

Продовження таблиці 4.2

1	2
7	Звук сирени швидкої допомоги, записаний на одній з центральних вулиць міста, в ясну погоду, вранці. Присутність інших звуків у записі досить помітна. Швидка проїжджає на досить великій відстані через сусіднє перехрестя.
8	Звук потягу, проїжджаючого повз.
9	Звук сигналізації автомобіля, записаний в дворі дому, увечері.
10	Звук, записаний на центральній вулиці міста. Чути звуки автомобілів, розмови людей, музику тощо.

Для тестування використовувався спеціальний модуль NodeMock, що імітує поведінку справжнього апаратного вузла системи, тобто приймає команди та реагує на них, зчитує аудіо-дані з файлу та відправляє їх у систему. Повний код модуля наведено у додатку І. Під час тестування у центральний модуль в якості залежності передається клас ConnectorMock, що наслідує базовий абстрактний клас ConnectorBase. Таким чином, центральний модуль не знає нічого про те, що він працює з тестовими даними і працює в стандартному режимі. ConnectorMock реалізує функціонал для роботи с екземплярами класу NodeMock, створює вузли, передає їм необхідні дані тощо (див. додаток И).

За результатами тестування було отримано наступні висновки:

– Звук сирени було розпізнано в 5 випадках з 7.
– Звук сирени не було розпізнано ні на одному з записів, що його не містять.

Результати тестування наведено у таблиці 4.3. Результати наведено у відсотках, що відповідає імовірності того, що у записі присутній звук сирени (звук сирени вважається розпізнаним, якщо імовірність > 85%):

Таблиця 4.3 – Результати тестування

№	Результат, %
1	89%
2	86%
3	91%
4	86%
5	88%
6	75%
7	63%
8	23%
9	44%
10	10%

Враховуючи результати тестування, ефективність алгоритму можна вважати задовільною. Звісно покладатися лише на розпізнавання звуку в такій важливій справі, як керування дорожнім рухом не правильно, тому є доцільним допрацювати систему, додавши розпізнавання світлових сигналів спеціального транспорту з використанням відео-камери.

4.3 Висновки

У даному розділі було детально описано процес тестування системи. Описано методологію модульного тестування. Було наведено основні принципи та підходи до модульного тестування. Було описано бібліотеки та модулі, що використовувалися при розробці модульних тестів для даної системи, зокрема пакет unittests, що призначений для реалізації модульного тестування та пакет dependency-injector – для реалізації DI. Також було наведено приклад коду, що призначений для тестування аудіо-буфера та результат його запуску.

Приведено опис процесу тестування алгоритму розпізнавання звуку. Описано основні принципи методології тестування білої скриньки, її переваги та

недоліки. Наведено список тест-кейсів, що використовувалися при тестуванні. Наведено алгоритм тестування та таблицю з результатами тестування. Зроблено відповідні висновки.

ЕКОНОМІЧНА ЧАСТИНА

Виконання будь-якої науково-дослідної та/або конструкторсько-технологічної роботи завжди вимагає певних витрат.

На основі економічних розрахунків можна довести економічну доцільність та ефективність впровадження отриманих результатів виконаних науково-дослідних робіт у виробництво, тобто здійснити так звану комерціалізацію наукових розробок. Саме цим завданням присвячено даний розділ магістерської кваліфікаційної роботи і передбачає він виконання таких етапів:

1. Оцінювання комерційного потенціалу розробки.
2. Розрахунок виробничої собівартості та ціни реалізації
3. Розрахунок комерційних ефектів та періоду окупності вкладених інвестицій

В рамках даного розділу буде розглянуто економічний потенціал розробки системи пріоритету екстреного транспорту. Дана система представляє собою комплект обладнання, що встановлюється на перехресті і за допомогою алгоритмів машинного навчання зменшує час проїзду екстрених транспортних засобів через перехрестя. Складовими компонентами системи є:

- 4 апаратних вузли для керування світлофорами, кожен з яких оснащений датчиками та пристроєм безпровідного зв’язку.
- Базова станція, що приймає інформацію від апаратних вузлів, обробляє її та відправляє назад команди.

Важливо зазначити, що у вищенаведеному списку розглядається комплект обладнання для типового перехрестя на 4 дороги, тоді як можливі і інші варіанти. Тож набір компонентів може змінюватися в залежності від ситуації.

5.1 Оцінювання комерційного потенціалу розробки

Для оцінки життєздатності продукту поміж іншого необхідно провести оцінку економічного потенціалу. Це допоможе нам визначити доцільність впровадження розробки у промислових масштабах.

Для найбільш об’єктивного оцінювання необхідно розглянути декілька блоків питань, таких як:

– Оцінка переваги для споживачів;
– Характеристики можливого ринку;
– Основні конкуренти;
– Здійсненність ідеї;
– Забезпеченість ресурсами.

Для оцінки комерційного потенціалу розробки було використано критерії, наведені в таблиці 5.1.

Таблиця 5.1 - Критерії оцінювання комерційного потенціалу розробки за 5-ти бальною шкалою

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
1	2	3	4	5	6
Технічна здійсненність концепції					
1	Достовір-ність концепції не підтверд-жена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахун-ками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження таблиці 5.1

1	2	3	4	5	6
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуата-ційні витрати значно вищі, ніж в аналогів	Експлуата-ційні витрати дещо вищі, ніж в аналогів	Експлуата-ційні витрати на рівні експлуатацій них витрат аналогів	Експлуата-ційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

Продовження таблиці 5.1

1	2	3	4	5	6
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінан-сування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінан-сування відсутні	Потрібні значні фінансові ресурси. Джерела фінан-сування є	Потрібні незначні фінансові ресурси. Джерела фінан-сування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використову-ються у військово-промис-ловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використо-вуються у виробництві

Продовження таблиці 5.1

1	2	3	4	5	6
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регла-ментних документів та отримання великої кількості дозвільних документів	Необхідно отримання великої кількості дозвільних документів, що вимагає значних коштів та часу	Процедура отримання дозвільних документів вимагає незначних коштів та часу	Необхідно тільки повідо-млення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 5.2.

Таблиця 5.2 - Результати оцінювання комерційного потенціалу розробки

Критерій	ПІБ експерта		
	Рейда О.М.	Майданюк В. П.	Денисюк А. В.
	Бали, виставлені експертами		
Технічна здійсненність концепції			
1	2	3	4

Продовження таблиці 5.2

1	2	3	4
Ринкові переваги (недоліки):			
2	4	4	3
3	3	4	3
4	2	3	3
5	3	2	2
Ринкові перспективи			
6	4	4	3
7	3	4	4
Практична здійсненність			
8	3	3	3
9	1	1	1
10	3	3	2
11	4	4	3
12	4	4	3
Сума балів	A_1=36	A_2=38	A_3=32

У таблиці 5.3 наведено рівні комерційного потенціалу [30]. Розрахуємо середню оцінку за даними таблиці 5.2 і порівняємо результат із заданими значеннями (формула 5.1).

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Сума балів	Рівень потенціалу
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

$$A_{\text{сер.}} = \frac{A_1+A_2+A_3}{3} = 35.33 \qquad (5.1)$$

Оцінка комерційного потенціалу становить 35.33, що відповідає рівню «Вище середнього».

5.2 Розрахунок виробничої собівартості та ціни реалізації

Виробнича собівартість містить витрати виробничого етапу. Ними є:

– Плата за сировину, основні виробничі матеріали та компоненти.
– Витрати на комунальні послуги та оренду офісу.
– Заробітна плата працівникам.
– Ремонт і утримання основних засобів.

Розрахуємо витрати на комплектуючі для розробки одного комплекту обладнання. Загальна формула для розрахунку вартості комплектуючих (формула 5.2) [30]:

$$E = \sum_1^n C_i * E_i * K_i \text{ [грн]}, \quad (5.2)$$

де C_i – кількість комплектуючих i-го виду, шт;

E_i – ціна комплектуючих i-го виду, грн;

K_i – коефіцієнт транспортних витрат. Надалі будемо вважати його рівним 15%.

Розрахуємо вартість компонентів одного апаратного вузла керування світлофором. В таблиці 5.4 наведено орієнтовний перелік компонентів, їх кількість та ціна.

Таблиця 5.4 – Компоненти апаратного вузла керування світлофором.

Компонент	Вартість, грн	Кількість, шт
1	2	3
Arduino Nano V3.0 AVR ATmega328P	700	1
Модуль мікрофона MAX9814	75	1
Модуль VGA камери OV7670	75	1

Продовження таблиці 5.4

1	2	3
Бездротовий модуль NRF24L01 з зовнішньою SMA антеною	80	1
Адаптер модуля NRF24L01	17	1
Корпус	100	1

Загальна вартість компонентів для виготовлення одного апаратного вузла $E_{\text{ав.}}$ становитиме (формула 5.3):

$$E_{\text{ав.}} = (700 + 75 + 75 + 80 + 17 + 100) * 1.15 = 1201[\text{грн}] \qquad (5.3)$$

Розрахуємо собівартість базової станції. В таблиці 5.5 наведено орієнтовний перелік компонентів, їх кількість та вартість.

Таблиця 5.5 – Компоненти базової станції.

Компонент	Вартість, грн	Кількість, шт
Raspberry Pi 4 Model B 8GB	4900	1
Бездротовий модуль NRF24L01 з зовнішньою SMA антеною	80	1
Адаптер модуля NRF24L01	17	1
Корпус	150	1

Загальна вартість компонентів базової станції $E_{\text{бс.}}$ становитиме (формула 5.4):

$$E_{\text{бс.}} = (4900 + 80 + 17 + 150) * 1.15 = 5919\ [\text{грн}] \qquad (5.4)$$

Розрахуємо вартість оснащення типового перехрестя (на 4 дороги). Оскільки на кожному перехресті буде знаходитись 4 апаратних вузли та 1 базова станція, вартість компонентів для виготовлення одного комплекту обладнання $E_{\text{к}}$ становитиме (формула 5.5):

$$E_{\text{к}} = E_{\text{бс.}} + E_{\text{ав.}} * 4 = 10723 \text{ [грн]} \quad (5.5)$$

Розрахуємо витрати на заробітну плату працівникам. Проведемо необхідні розрахунки. У таблиці 5.6 наведено орієнтовний перелік працівників, їх кількість та заробітна плата:

Таблиця 5.6 – Перелік працівників, їх кількість та заробітна плата.

Працівник	Кількість, шт	ЗП, грн/міс
Інженер апаратного забезпечення	4	25000
Інженер програмного забезпечення	2	60000
Тестувальник	2	15000
Працівник	Кількість, шт	ЗП, грн/міс
Менеджер з продажів	2	15000
Директор	1	10000

Додаткову заробітну плату співробітника будем вважати рівною 10 відсоткам. Врахуємо також єдиний соціальний внесок, що становить 22%.

Таким чином загальна сума витрат на заробітну плату працівникам $E_{\text{зп.}}$ в місяць становитиме (формула 5.6) [30]:

$$E_{\text{зп.}} = (4 * 25000 + 2 * 60000 + 15000 + 15000 + 10000) * 1.1 * 1.22 = 389180 \text{ [грн/міс]} \quad (5.6)$$

Розрахуємо загальновиробничі витрати, куди входять оренда офісу, оплата комунальних послуг та витратних матеріалів: канцелярії, їжі, засобів гігієни тощо. В таблиці 5.7 наведено орієнтовну вартість.

Таблиця 5.7 – Орієнтовний перелік помісячних витрат на оренду, комунальні послуги та витратні матеріали.

Назва	Вартість, грн/міс
Оренда офісу	15000
Комунальні послуги	10000
Витратні матеріали	5000

Врахуємо також додаткові неочікувані витрати. Для цього введемо коефіцієнт додаткових витрат, що буде дорівнювати десяти відсоткам. Таким чином, загальна вартість оренди та обслуговування офісу $E_{\text{зв.}}$ Становитиме (формула 5.7) [30]:

$$E_{\text{зв.}} = (15000 + 10000 + 5000) * 1.1 = 33000\ [\text{грн/міс}] \qquad (5.7)$$

Розрахуємо початкові разові витрати на закупки основних засобів, куди входить: обладнання, меблі, необхідні інструменти тощо. Орієнтовний перелік наведено у таблиці 5.8.

Таблиця 5.8 – Орієнтовний перелік обладнання та меблів необхідних для початку роботи.

Назва	Кількість, шт	Ціна, грн	Вартість, грн
Стіл робочий	11	4000	44000
Крісло робоче	11	4000	44000
Інструменти	8	2000	16000
Ноутбук	7	20000	140000
Загальна вартість	244000		

Таким чином орієнтовна вартість основних засобів $E_{\text{оз.}}$ Становитиме (формула 5.8) [30]:

$$E_{\text{оз.}} = 44000 * 2 \ + 16000 + 140000 = 244000 \text{ [грн]} \quad (5.8)$$

Для того, щоб врахувати у виробничій собівартості значення витрат на основні засоби, необхідно розрахувати амортизаційні відрахування. Для розрахунку будемо використовувати формулу лінійної амортизації. Лінійна амортизація - це такий різновид списання вартості основних засобів, при якій щорічне зменшення їх вартості протягом усього періоду списання залишається одним і тим же (формула 5.9) [31]. Отже:

$$E_{\text{ам.}} = \frac{E_{\text{оз.}}}{T} \text{ [грн/міс]}, \quad (5.9)$$

де Т – період списання, Т = 12 [міс].

$E_{\text{ам.}} = \frac{244000}{12} = 20333$ [грн/міс]

Розрахуємо виробничу собівартість за формулою 5.10:

$$E_{\text{вс.}} = \frac{E_{\text{ам.}} + E_{\text{зп.}} + E_{\text{зв.}}}{N} + E_{\text{к}} \text{ [грн/міс]}, \quad (5.10)$$

де N – кількість випущених комплектів за місяць, N = 400 [шт].

$E_{\text{вс.}} = \frac{20333 + 389180 + 33000}{400} + 10723 = 11829$ [грн/міс]

Розрахуємо ціну реалізації виробу $S_{\text{р}}$ за формулою 5.11 [30]:

$$S_{\text{р}} = E_{\text{вс.}} * p * w \text{ [грн]}, \quad (5.11)$$

де $E_{\text{вс.}}$ – значення виробничої собівартості;

p – коефіцієнт рентабельності, p = 1.3;

w – коефіцієнт податку на додану вартість, w = 1.2.

$S_{\text{р}} = 11829 * 1.3 * 1.2 = 18453$ [грн]

5.3 Розрахунок комерційних ефектів та періоду окупності вкладених інвестицій

Для оцінки комерційного ефекту від реалізації результатів розробки, необхідно визначити чистий прибуток, що підприємство буде отримувати кожного місяця. Чистий прибуток можна розрахувати за формулою 5.12:

$$I = (S_{\text{р}} - E_{\text{вс.}}) * N \text{ [грн/міс]}, \quad (5.12)$$

де $S_{\text{р}}$ – значення ціни реалізації;

N – кількість комплектів обладнання, реалізованих за місяць;

Для розрахунку терміну окупності, розглянемо три сценарії:

- Оптимістичний – продано 30 комплектів обладнання на місяць.
- Реалістичний – продано 15 комплектів обладнання на місяць.
- Песимістичний – продано 10 комплектів обладнання на місяць.

Розрахуємо місячний прибуток за кожним з можливих сценаріїв.

За оптимістичним прогнозом:

$I_{\text{опт.}} = (18453 - 11829) * 30 = 198720$ [грн/міс]

За реалістичним прогнозом:

$I_{\text{рс.}} = (18453 - 11829) * 20 = 132480$ [грн/міс]

За песимістичним прогнозом:

$I_{\text{пс.}} = (18453 - 11829) * 10 = 66240$ [грн/міс]

Для оцінки окупності вкладених інвестицій, розрахуємо кількість інвестицій, що потрібні для початку роботи. Зробити це можна, додавши значення загальновиробничих витрат до значення витрат на закупку основних засобів. Отже кількість початкових інвестицій становитиме (формула 5.13):

$$E_{\text{п}} = E_{\text{оз.}} + E_{\text{зв.}} = 244000 + 33000 = 277000 \text{ [грн]} \qquad (5.13)$$

Для оцінки окупності вкладених інвестицій зведемо дані отримані у попередніх підрозділах у три відповідні таблиці. У таблиці 5.9 наведено дані, що відповідають оптимістичному прогнозу, у таблицях 5.10 та 5.11 – реалістичному та песимістичному відповідно.

Таблиця 5.9 – Прибутки за оптимістичним прогнозом

Місяць	Прибуток, грн/міс	Сумарний прибуток, грн
1	198720	-78280
2	198720	120440
3	198720	319160
4	198720	517880
5	198720	716600
6	198720	915320
7	198720	1114040
8	198720	1312760
9	198720	1511480
10	198720	1710200
11	198720	1908920
12	198720	2107640

Таблиця 5.10 – Прибутки за реалістичним прогнозом

Місяць	Прибуток, грн/міс	Сумарний прибуток, грн
1	132480	-144520
2	132480	-12040
3	132480	120440
4	132480	252920
5	132480	385400
6	132480	517880
7	132480	650360
8	132480	782840
9	132480	915320
10	132480	1047800
11	132480	1180280
12	132480	1312760

Таблиця 5.11 – Прибутки за песимістичним прогнозом

Місяць	Прибуток, грн/міс	Сумарний прибуток, грн
1	2	3
1	66240	-210760
2	66240	-144520
3	66240	-78280
4	66240	-12040
5	66240	54200
6	66240	120440

Продовження таблиці 5.11

1	2	3
7	66240	186680
8	66240	252920
9	66240	319160
10	66240	385400
11	66240	451640
12	66240	517880

5.4 Висновки

На рисунку 5.1 відображено дані окупності за трьома прогнозами. Бачимо, що за найоптимістичнішими очікуваннями, вкладені інвестиції повернуться вже після першого місяця роботи. За реалістичним прогнозом – термін окупності складає близько двох місяців. У найгіршому ж випадку – близько чотирьох місяців.

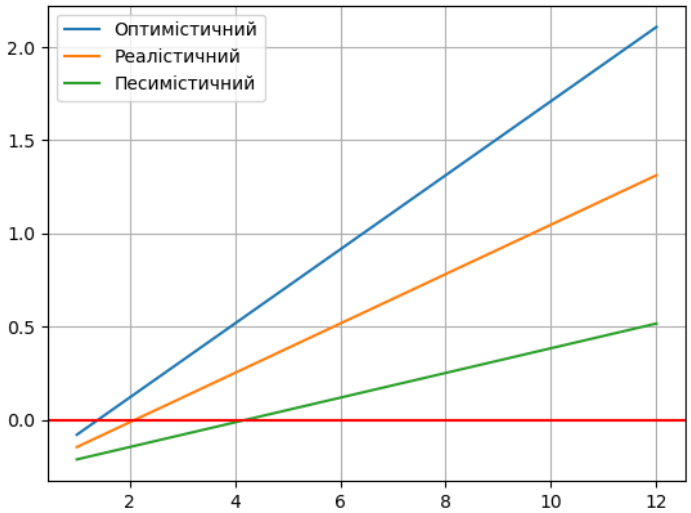


Рисунок 5.1 – Графік окупності вкладених інвестицій за оптимістичним, реалістичним та песимістичним прогнозами

Враховуючи результати розрахунків, можна зробити висновок, що потенціал даної розробки є досить великим. Термін окупності є досить малим навіть за найгіршими прогнозами.

Важливо зазначити, що всі наведені в даному розділі розрахунки є приблизними, а списки компонентів, матеріалів та ін. є орієнтовними та можуть і будуть змінюватися в подальшому.

Також, є великий потенціал до зменшення виробничої собівартості обладнання за рахунок зміни компонентів. Наведені в цьому розділі компоненти були обрані так, щоб можна було отримати працюючий пристрій у якомога менші строки. В подальшому, коли процес розробки стабілізується, можливо змінювати компоненти на більш дешеві аналоги, не виключаючи і можливості розробки власних компонентів.

ВИСНОВКИ

У рамках даної магістерської кваліфікаційній роботи було розроблено систему управління пріоритетним перетином перехрестя транспортних засобів екстрених служб. У роботі:

- Проведено аналіз існуючих методів і засобів контролю світлофорів для проїзду спеціального транспорту.
- Проведено детальний аналіз існуючих методів і засобів контролю світлофорів для проїзду спеціального транспорту.
- Запропоновано метод виявлення спеціальних транспортних засобів на основі розпізнавання звуку сирени.
- На основі даних аналізу та запропонованого методу було спроектовано систему пріоритетного проїзду екстрених транспортних засобів. Було розроблено проектну документацію, визначено основні складові частини та компоненти.
- На основі проекту було розроблено систему для розпізнавання наявності спеціального транспорту та контролю світлофорів на перехресті.
- Проведено тестування та доведено працездатність системи.

В першому розділі було детально розглянуто та проаналізовано 27 робіт. Було визначено основні методи та підходи до створення таких систем управління пріоритетним перетином перехрестя. Було проаналізовано сильні і слабкі сторони кожної з них, та зроблено відповідні висновки.

Також було розглянуто природу звукових хвиль, загальні підходи та методи в задачах розпізнавання звуку та основні проблеми, які можуть виникати в процесі розробки системи розпізнавання звуку. Було розглянуто та проаналізовано основні підходи до задач розпізнавання звуку.

Було виконано постановку задачі, визначено основні завдання та етапи роботи.

У другому розділі було проаналізовано та обґрунтовано вибір мови програмування. Було розглянуто основні середовища розробки та обґрунтовано вибір.

Проведено аналіз основних інструментальних засоби, що використовувалися для розробки. Крім цього було наведено основні бібліотеки, що використовувалися в ході роботи та опис їх методів.

Приведено базову схему роботи системи. Було вказано основні принципи дії та складові компоненти, а також базовий алгоритм роботи.

Також було детально розглянуто запропонований алгоритм розпізнавання звуку. Були наведені основні формули та розрахунки. Також було описано спосіб фільтрації. Було обґрунтовано вибір фільтра Баттерворта у якості смугового фільтра для використання в роботі.

Було проведено концептуальне проектування системи, розроблено діаграму послідовності, що відображає кроки алгоритму та створено діаграму, що показує основні модулі системи та зв’язки між ними.

Було наведено опис програмної реалізації. Приведено фрагменти коду основних складових частин. Було розроблено та продемонстровано діаграму потоку даних, з якої видно основні програмні модулі та зв’язки між ними. Окрім цього було розглянуто загальну структуру проекту та її складові. Було детально описано всі пакети, з яких складається проект, їх складові та призначення.

У четвертому розділі було детально описано процес тестування системи. Зокрема було описано методологію модульного тестування. Також було наведено основні принципи та підходи до модульного тестування. Було описано бібліотеки та модулі, що використовувалися при розробці модульних тестів для даної системи, зокрема пакет unittests, що призначений для реалізації модульного тестування та пакет dependency-injector – для реалізації DI. Також було наведено приклад коду, що призначений для тестування аудіо-буфера та результат його запуску.

Приведено опис процесу тестування алгоритму розпізнавання звуку. Описано основні принципи методології тестування білої скриньки, її переваги та недоліки. Наведено список тест-кейсів, що використовувалися при тестуванні. Наведено алгоритм тестування та таблицю з результатами тестування. Зроблено відповідні висновки.

В останньому розділі було проаналізовано економічну доцільність розробки. Було проведено оцінювання комерційного потенціалу розробки. Також було розраховано виробничу собівартість та орієнтовну ціну реалізації. На основі розрахунків, було проведено прогноз комерційних ефектів та періоду окупності розробки. Результат був визнаний задовільним.

У процесі досліджень використовувались: теорія сигналів, теорія диференціально-інтегрального числення, гармонічний аналіз, функціональний аналіз. Комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

ПЕРЕЛІК ПОСИЛАНЬ

1. Устінова О. Чому ти можеш збити "швидку" і тобі за це нічого не буде? IEEE: Українська газета «Правда», 2021. [Електронний ресурс] URL.: https://blogs.pravda.com.ua/authors/ ustinova/6024e96ed59 95/

2. Міністерство охорони здоров'я України. Надання екстреної медичної допомоги потребує вдосконалення: результати незалежної оцінки якості, 2019. [Електронний ресурс] URL.: https://moz.gov.ua/article/news/ nadannja-ekstrenoi-medichnoi-dopomogi-potrebue-vdoskonalennja-rezultati-nezalezhnoi-ocinki-jakosti

3. Mehendale N., Patel R., Mange S., Mulik S. Review of Emergency Vehicle Priority Systems. IEEE: SSRN, 2021. [Електронний ресурс] URL.: https:// deliverypdf.ssrn.com/delivery.php?ID=864126029100103104026009089072113095042023056058091028102122005119000071076106085027025117016012040001016121066123065115084003030022087053013099114125012085013019027072010003110107083009027116065114064090106101102068119097099115109082125087091127011079&EXT=pdf&INDEX=TRUE

4. Tran, V.T., Tsai, W.H. Acoustic-based emergency vehicle detection using convolutional neural networks. IEEE Access 8, 2020. pp. 75702–75713.

5. Ellis, D.P.: Detecting alarm sounds, 2001.

6. Beritelli F., Casale S., Russo A., Serrano S. An automatic emergency signal recognition system for the hearing impaired, IEEE 12th Digital Signal Processing. Workshop & 4th IEEE Signal Processing Education Workshop, 2006. pp. 179–182.

7. Carbonneau M.A., Lezzoum N., Voix J., Gagnon G. Detection of alarms and warning signals on an digital in-ear device. International Journal of Industrial Ergonomics 43(6), 2013. pp. 503–511.

8. Clontz S., Adhami R. Long-duration signal detection in a noisy environment. Proceedings. IEEE: The Twenty-First Southeastern Symposium on System Theory, 1989. pp. 527–532.

9. Carmel D., Yeshurun A., Moshe Y. Detection of alarm sounds in noisy environments. IEEE: 25th European Signal Processing Conference (EUSIPCO), 2017. pp. 1839–1843.

10. Brill, W.E. Emergency vehicle detection system. IEEE: US Patent, 2002. pp. 6, 362, 749.

11. Meucci F., Pierucci L., Del Re E., Lastrucci L., Desii P. A real-time siren detector to improve safety of guide in traffic environment. IEEE: 16th European Signal Processing Conference, 2008. pp. 1–5.

12. Shibuya S., Yoshida T., Yamashiro Z., Miyawaki M. Fast emergency vehicle preemption systems. IEEE: Transportation research record, 2000. pp. 1739(1), 44–50.

13. Bharadwaj R., Deepak J., Baranitharan M., Vaidehi V. Efficient dynamic traffic control system using wireless sensor networks. IEEE: International Conference on Recent Trends in Information Technology (ICRTIT), 2013. pp. 668–673.

14. Masum, A. K. M., Chy M. K. A., Rahman I., Uddin M. N., Azam, K. I. An internet of things (iot) based smart traffic management system: a context of bangladesh. IEEE: International Conference on Innovations in Science, Engineering and Technology (ICISET), 2018. pp. 418–422.

15. Kodire V., Bhaskaran S., Vishwas H.: Gps and zigbee based traffic signal preemption. IEEE: International Conference on Inventive Computation Technologies (ICICT), vol. 2, 2016. pp. 1–5.

16. Eltayeb A. S., Almubarak H. O., Attia T. A. A gps based traffic light premption control system for emergency vehicles. IEEE: 2013 International conference on computing, electrical and electronic engineering (ICCEEE), 2013. pp. 724–729.

17. Buchenscheit A., Schaub F., Kargl F., Weber M. A vanet-based emergency vehicle warning system. IEEE: Vehicular Networking Conference (VNC), 2009. pp. 1–8.

18. Younes M. B., Boukerche A. An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems. IEEE: Wireless Networks 24(7), 2018 pp. 2451–2463.

19. Shaaban K., Khan M. A., Hamila R., Ghanim M.: A strategy for emergency vehicle preemption and route selection. IEEE: Arabian Journal for Science and Engineering, 2019. pp. 44(10), 8905–8913.

20. Bygrave C., Dineen O., Norton P., Bell D. R. Early warning system for approaching emergency vehicles. IEEE: US Patent, 2009. pp. 7, 515, 065.

21. Sundar R., Hebbar S., Golla V. Implementing intelligent traffic control system for congestion control, ambulance clearance, and stolen vehicle detection. IEEE: Sensors Journal, 2014. pp. 15(2), 1109–1113.

22. Masoud M., Belkasim S. Wsn-evp: A novel special purpose protocol for emergency vehicle preemption systems. IEEE: Transactions on Vehicular Technology, 2017. pp. 67(4), 3695–3700.

23. Huang Y. S., Weng Y. S., Zhou M. Design of traffic safety control systems for emergency vehicle preemption using timed petri nets. IEEE: Transactions on Intelligent Transportation Systems, 2015. pp. 16(4), 2113–2120.

24. Mirchandani P. B., Lucas D. E. Integrated transit priority and rail/emergency preemption in real-time traffic adaptive signal control. IEEE: Journal of Intelligent Transportation Systems, 2004. pp.8(2), 101–115.

25. Louati A., Elkosantini S., Darmoul S., Louati H. Multi-agent preemptive longest queue first system to manage the crossing of emergency vehicles at interrupted intersections. IEEE: European Transport Research Review, 2018 pp. 10(2), 1–21.

26. Abdoos M., Mozayani N., Bazzan A.L. Traffic light control in non-stationary environments based on multi agent q-learning. IEEE: 2011 14th International IEEE conference on intelligent transportation systems (ITSC), 2011. pp. 1580–1585.

27. Gokulan B. P., Srinivasan D. Distributed geometric fuzzy multiagent urban traffic signal control. IEEE: Transactions on Intelligent Transportation Systems, 2010, pp.11(3), 714–727.

28. Binish F., Preethi A., Hrushikesh V., Akhilesh Singh B.; Harshanikethan R. Kotion. An automatic siren detection algorithm using Fourier Decomposition Method and MFCC. IEEE: **Xplore,** 2020. [Електронний ресурс] URL.: https://ieeexplore. ieee.org/document/9225414

29. Butterworth S. On the Theory of Filter Amplifiers IEEE: Admiralty research laboratory, 1930. [Електронний ресурс] URL.: https://www.changpuak. ch/electronics/downloads/On_the_Theory_of_Filter_Amplifiers.pdf

30.Козловський В. О. Техніко-економічні обґрунтування та економічні розрахунки в дипломних проектах та роботах. IEEE: ВДТУ, 2021. [Електронний ресурс] URL.: http://epvm.vntu.edu.ua/wp-content/uploads/2021/09/Ekonom_dyplom_2003.pdf

31. Методи амортизації основних засобів (ОЗ) IEEE: Інформаційно-консультаційна платформа «zakon.help», 2017. [Електронний ресурс] URL.: https://zakon.help/article/metodi-amortizacii-osnovnih-zasobiv-oz

32. Методы тестирования программного обеспечения. IEEE: БЛОГ Web Программиста, 2016. [Електронний ресурс] URL.: http://juice-health.ru/program/software-testing/495-software-testing-methods

33. Модульное тестирование: что это? Типы, инструменты. IEEE: logrocon, 2021. [Електронний ресурс] URL.: https://logrocon.ru/news/unit_testing

ДОДАТОК А.
ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

« 13 » вересня 2021 р.

Технічне завдання

на магістерську кваліфікаційну роботу

«Програмна система передчасного попередження про рух спеціального транспорту на перехресті»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

Рейда О. М., к.т.н., доцент кафедри ПЗ

"____"_____2021 р.

Виконав:

студент гр.2ПІ-20м Подобрій В. І.

"____"_____2021 р.

Вінниця – 2021 рік

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Програмна система передчасного попередження про рух спеціального транспорту на перехресті».

Галузь застосування – системи ПЕТЗ.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ ректора по ВНТУ № 277 від « 24 » вересня 2021 р. про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є покращення контролю світлофорів на перехресті з метою забезпечення пріоритетного проїзду спеціального транспорту: автомобілів швидкої допомоги, поліцейських та пожежних.

Призначення роботи – розробка методів і засобів передчасного попередження про рух спеціального транспорту на перехресті.

3 Вихідні дані для проведення МКР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1. Основы цифровой обработки сигналов: Курс лекций /Авторы: А. И. Солонина, Д. А. Улахович, С. М. Арбузов, Е. Б. Соловьева/ Изд. 2-е испр. и перераб. - СПб.: БХВ-Петербург, 2005. - 768 с.

2. Айфичер, Эммануил С., Джервис, Барри У. Цифровая обработка сигналов: практический подход, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 992 с.

3. Рабинер Л., Гоулд Б. Теория и применение цифровой обработки сигналов: Пер. с англ. – М.: Мир, 1978. – 500 с.

4. Технічні вимоги

Стандартизовані звукові сигнали для систем ПЕТЗ, параметри сигналів систем ПЕТЗ, звукові файли із сигналами систем ПЕТЗ, вимоги до системи визначення звукового сигналу (наявність датчиків звуку по всіх напрямках перехрестя), використання автономної системи реагування на сигнали ПЕТЗ.

5. Конструктивні вимоги.

Розробка повинна відповідати усім функціональним вимогам. Повинна бути зручною в використанні та налагодженні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації та ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз методів розпізнавання звуку та побудови систем екстрених транспортних засобів	15.09.2021-26.09.2021
2	Розробка методу розпізнавання звуку	27.09.2021-15.10.2021
3	Проектування системи пріоритету екстрених транспортних засобів	16.10.2021-7.11.2021
4	Програмна реалізація системи пріоритету екстрених транспортних засобів	8.11.2021-21.11.2021
5.	Економічна частина	22.11.2021-30.11.2021

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б.
ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

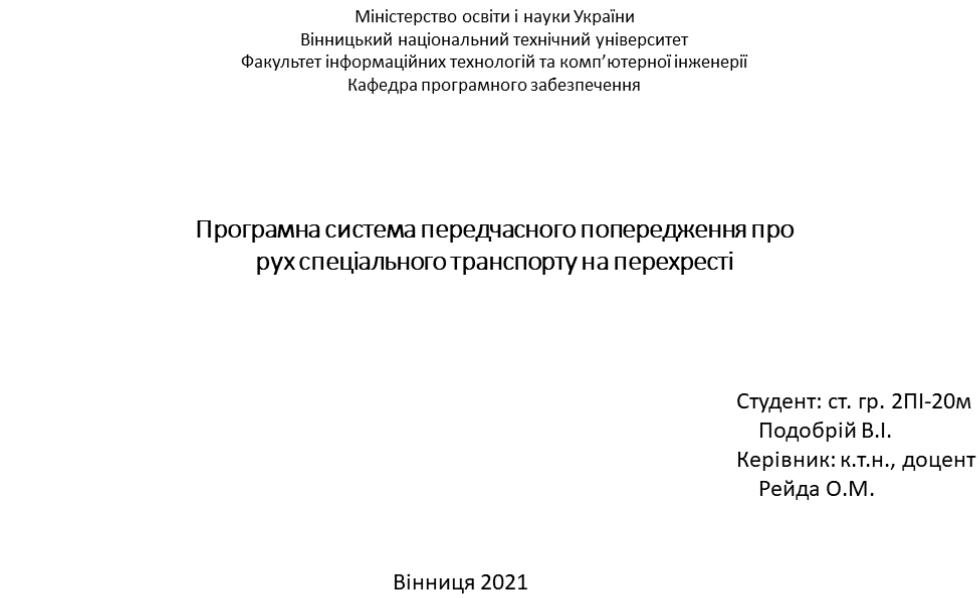


Рисунок Б.1. Плакат №1

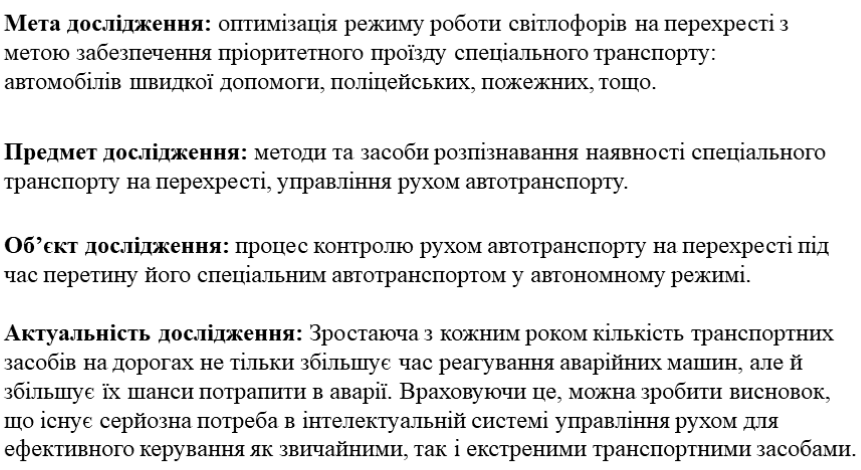


Рисунок Б.2. Плакат №2

Основні задачі дослідження

- Провести аналіз існуючих методів і засобів контролю світлофорів для проїзду спеціального транспорту.
- Запропонувати метод розпізнавання наявності спеціального транспорту на перехресті.
- На основі запропонованого методу розробити систему для розпізнавання наявності спеціального транспорту та контролю світлофорів на перехресті.
- Провести експериментальні випробування розробленої системи.

Рисунок Б.3. Плакат №3

Наукова новизна

- Розроблено класифікацію систем пріоритету екстрених транспортних засобів (ПЕТЗ).
- Подальшого розвитку отримав метод визначення звуку на основі фільтрування та амплітудного аналізу.
- Запропоновано архітектуру системи пріоритетного перетину перехрестя транспортними засобами екстрених служб.

Практична цінність

На основі розробки, що розглядається в даній роботі, можливо побудувати систему, що здатна значно зменшити час проїзду екстрених транспортних засобів через перехрестя, що в свою чергу зменшить час прибуття до місця призначення.

Рисунок Б.4. Плакат №4

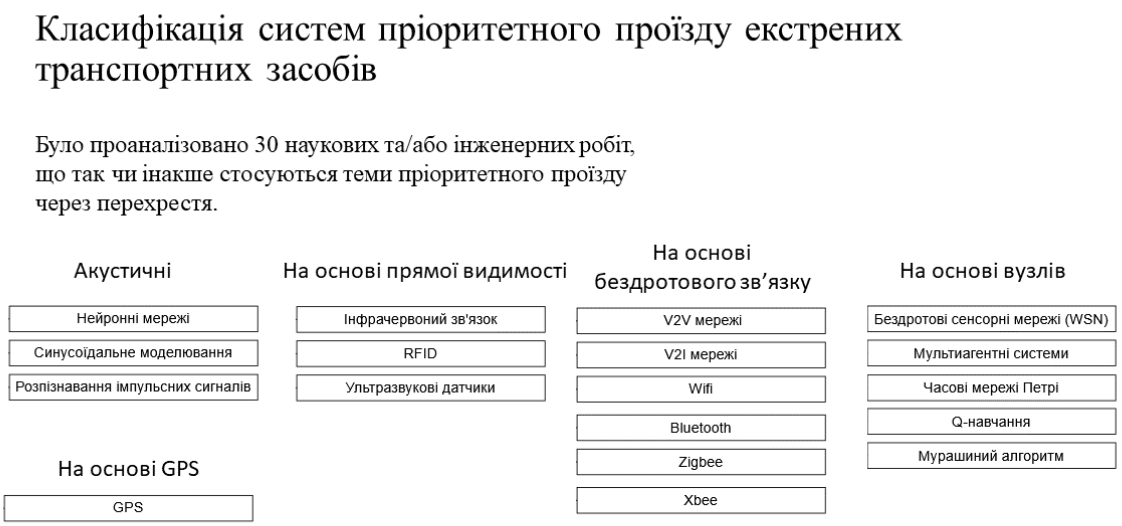


Рисунок Б.5. Плакат №5

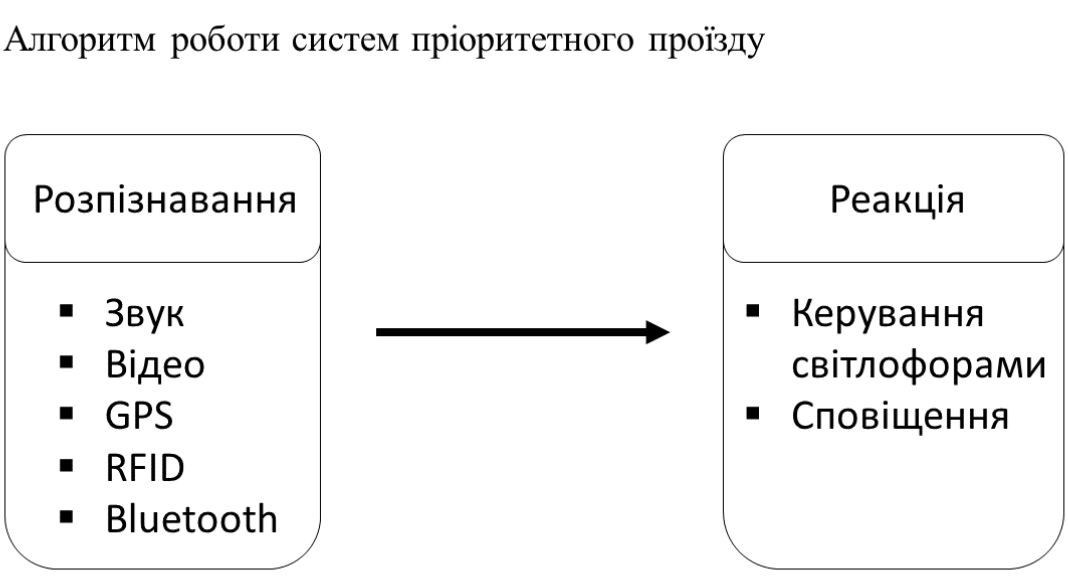
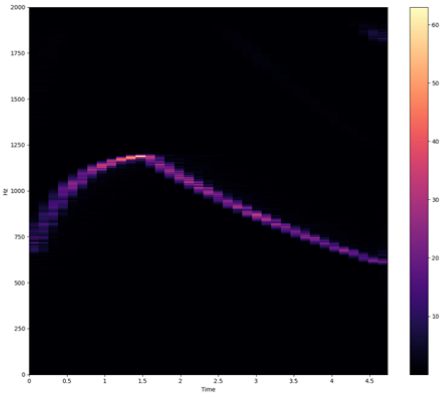


Рисунок Б.6. Плакат №6

Опис методу розпізнавання звуку

1. Відфільтрувати вхідний аудіо-сигнал смуговим фільтром або декількома фільтрами
2. Порівняти амплітуду вихідного сигналу з певним пороговим значенням
3. Прийняти рішення



Спектрограма еталонного звуку сирени

Рисунок Б.7. Плакат №7

Алгоритм фільтрації

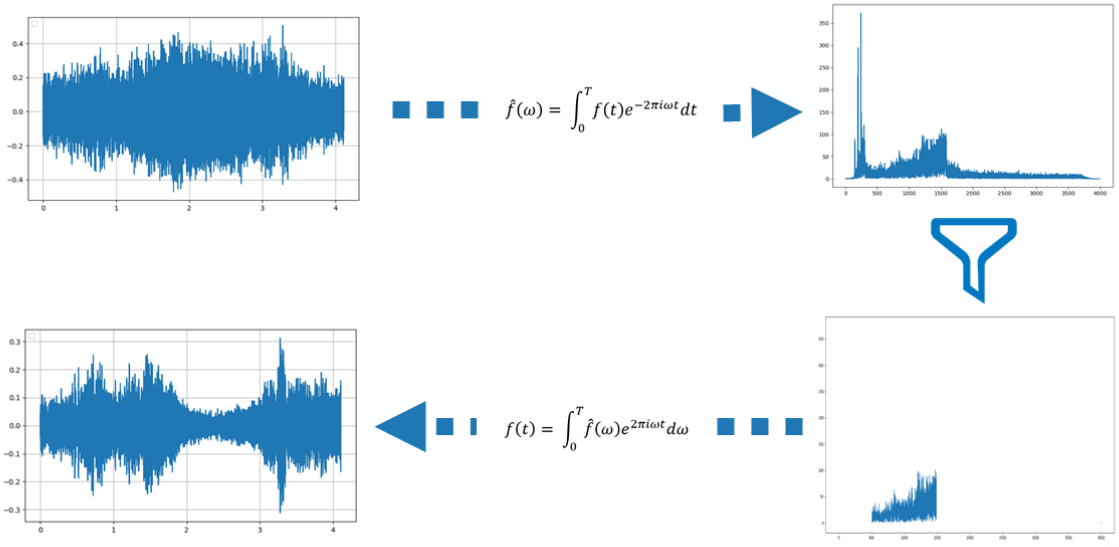
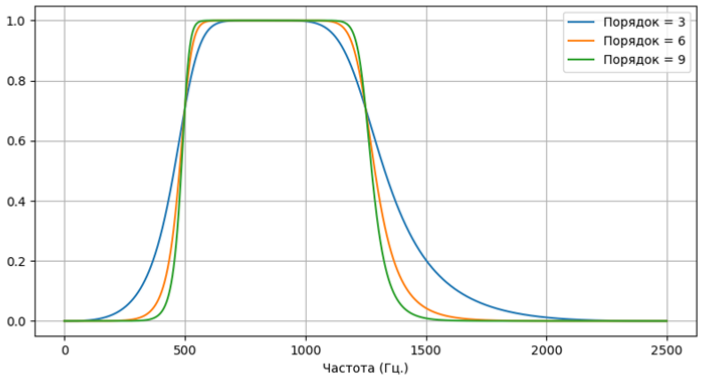
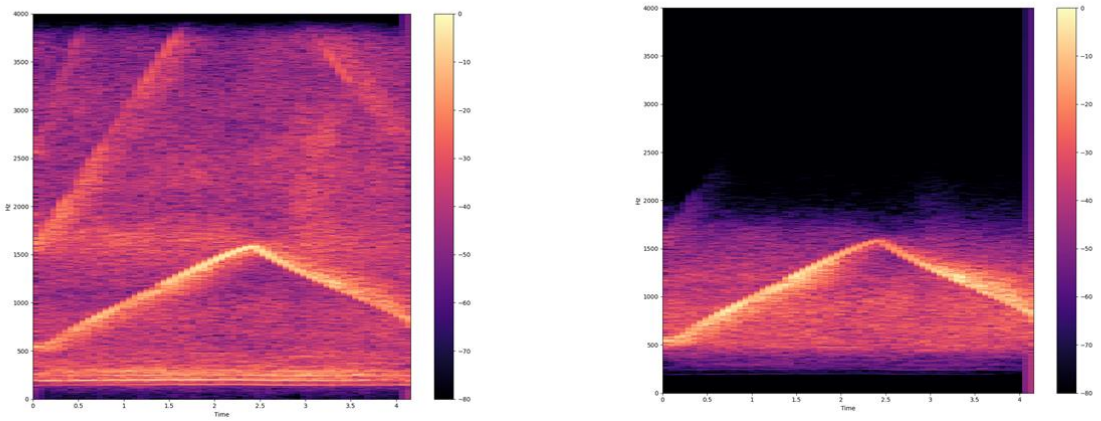


Рисунок Б.8. Плакат №8



Частотний відклик фільтру Баттерворта 3-го, 6-го та 9-го порядків

Рисунок Б.9. Плакат №9



Спектрограма початкового (ліворуч) та відфільтрованого (праворуч) звуку сирени

Рисунок Б.10. Плакат №10

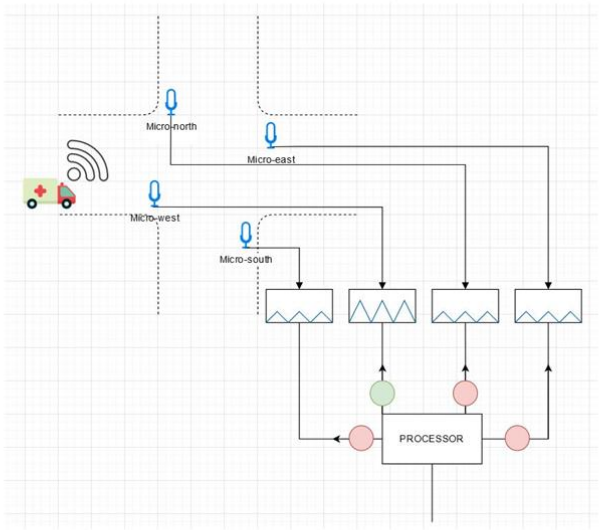


Рисунок Б.11. Плакат №11

Діаграма потоку даних

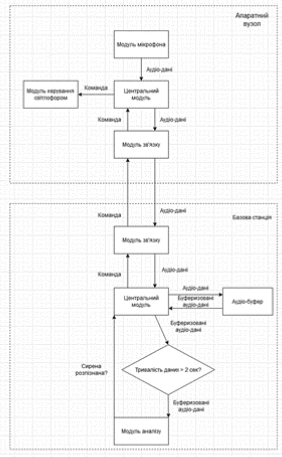
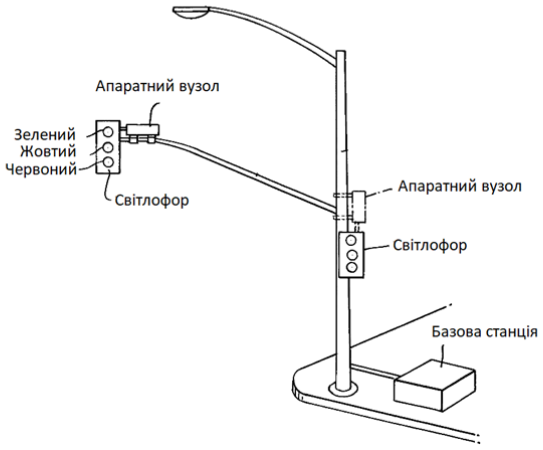


Рисунок Б.12. Плакат №12



2 варіанти можливого розташування компонентів системи

Рисунок Б.13. Плакат №13

Тестування системи

Для тестування було зроблено сім записів зі звуком сирени автомобіля швидкої допомоги в різних частинах міста, зокрема при різних погодних умовах, з різної відстані, для автомобіля, що находиться в русі або стоїть на місці тощо. Крім цього було записано інші звуки: шуми дорожнього руху, музику з придорожніх магазинів, звук потягу тощо.

За результатами тестування було отримано наступні висновки:

- Звук сирени було розпізнано в 5 випадках з 7.
- Звук сирени не було розпізнано ні на одному з записів, що його не містять.

Рисунок Б.14. Плакат №14

Висновки

В рамках роботи було виконано наступні завдання:

- Обгрунтовано доцільність дослідження.
- Проаналізовано існуючі підходи в задачі пріоритетного проїзду через перехрестя.
- На основі аналізу розроблено класифікацію систем пріоритету екстрених транспортних засобів.
- Подальшого розвитку отримав метод визначення звуку на основі фільтрування та амплітудного аналізу.
- На основі запропонованого методу було спроектовано системи пріоритету екстрених транспортних засобів.
- На основі проектної документації було виконано програмну реалізацію.
- Було проведено тестування системи та доведено її працездатність
- Було обгрунтовано економічну доцільність розробки

Рисунок Б.15. Плакат №15

ДЯКУЮ ЗА УВАГУ

Рисунок Б.16. Плакат №16

ДОДАТОК В.
ЛІСТИНГ РЕАЛІЗАЦІЇ ФІЛЬТРАЦІЇ З ДОПОМОГОЮ ФІЛЬТРУ БАТТЕРВОРТА

```
from scipy.signal import butter, lfilter

class BandpassFilter:
    def __init__(self, low, high, sampleRate, order=5, ftype="band"):
        self.low = low
        self.high = high
        self.sampleRate = sampleRate
        self.order = order
        self.ftype = ftype

        maxFr = self.sampleRate / 2
        self.__lowNorm = low / maxFr
        self.__highNorm = high / maxFr

    def filter(self, data):
        b, a = butter(self.order, [self.__lowNorm, self.__highNorm], self.ftype)
        return lfilter(b, a, data)
```

ДОДАТОК Г.
ЛІСТИНГ ПРОГРАМИ ДЛЯ НАДСИЛАННЯ СИГНАЛІВ ЧАСТОТОЮ 433 МГЦ

```
import argparse
import logging

from rpi_rf import RFDevice

logging.basicConfig(level=logging.INFO, datefmt='%Y-%m-%d %H:%M:%S',
                    format='%(asctime)-15s - [%(levelname)s] %(module)s:
%(message)s',)

parser = argparse.ArgumentParser(description='Sends a decimal code via a
433/315MHz GPIO device')
parser.add_argument('code', metavar='CODE', type=int,
                    help="Decimal code to send")
parser.add_argument('-g', dest='gpio', type=int, default=17,
                    help="GPIO pin (Default: 17)")
parser.add_argument('-p', dest='pulselength', type=int, default=None,
                    help="Pulselength (Default: 350)")
parser.add_argument('-t', dest='protocol', type=int, default=None,
                    help="Protocol (Default: 1)")
parser.add_argument('-l', dest='length', type=int, default=None,
                    help="Codelength (Default: 24)")
parser.add_argument('-r', dest='repeat', type=int, default=10,
                    help="Repeat cycles (Default: 10)")
args = parser.parse_args()

rfdevice = RFDevice(args.gpio)
rfdevice.enable_tx()
```

```
rfdevice.tx_repeat = args.repeat

if args.protocol:
    protocol = args.protocol
else:
    protocol = "default"
if args.pulselength:
    pulselength = args.pulselength
else:
    pulselength = "default"
if args.length:
    length = args.length
else:
    length = "default"

logging.info(str(args.code) +
             " [protocol: " + str(protocol) +
             ", pulselength: " + str(pulselength) +
             ", length: " + str(length) +
             ", repeat: " + str(rfdevice.tx_repeat) + "]")

rfdevice.tx_code(args.code, args.protocol, args.pulselength, args.length)
rfdevice.cleanup()
```

ДОДАТОК Д.
ЛІСТИНГ ПРОГРАМИ ДЛЯ ОТРИМАННЯ СИГНАЛІВ ЧАСТОТОЮ 433 МГЦ

```
import argparse
import signal
import sys
import time
import logging

from rpi_rf import RFDevice

rfdevice = None

# pylint: disable=unused-argument
def exithandler(signal, frame):
    rfdevice.cleanup()
    sys.exit(0)

logging.basicConfig(level=logging.INFO, datefmt='%Y-%m-%d %H:%M:%S',
                    format='%(asctime)-15s - [%(levelname)s] %(module)s:
%(message)s', )

parser = argparse.ArgumentParser(description='Receives a decimal code via a
433/315MHz GPIO device')
parser.add_argument('-g', dest='gpio', type=int, default=27,
                    help="GPIO pin (Default: 27)")
args = parser.parse_args()

signal.signal(signal.SIGINT, exithandler)
rfdevice = RFDevice(args.gpio)
```

```
rfdevice.enable_rx()
timestamp = None
logging.info("Listening for codes on GPIO " + str(args.gpio))
while True:
    if rfdevice.rx_code_timestamp != timestamp:
        timestamp = rfdevice.rx_code_timestamp
        logging.info(str(rfdevice.rx_code) +
                     " [pulselength " + str(rfdevice.rx_pulselength) +
                     ", protocol " + str(rfdevice.rx_proto) + "]")
    time.sleep(0.01)
rfdevice.cleanup()
```

ДОДАТОК Е.
ЛІСТИНГ КОДУ АУДІО-БУФЕРА

```
import numpy as np

class AudioBuffer:
    def __init__(self, sampleRate):
        self.__sampleRate = sampleRate
        self.__audioData = np.zeros(0)

    def duration(self):
        return self.length() / self.__sampleRate

    def length(self):
        return len(self.__audioData)

    def getData(self):
        return self.__audioData

    def append(self, data):
        self.__audioData = np.append(self.__audioData, data)

    def clear(self):
        self.__audioData = np.zeros(0)
```

ДОДАТОК Ж.
ЛІСТИНГ МОДУЛЯ РОЗПІЗНАВАННЯ СИРЕНИ

```
from filters.BandpassFilter import BandpassFilter

class AmbulanceRecognizer:
    def __init__(self, settings):
        self.__filter = BandpassFilter(500, 1250, settings.sampleRate)
        self.__treshhold = 10

    def recognizeAudio(self, data):
        filteredData = self.__filter.filter(data)
        if filteredData.sum() / len(filteredData) > self.__treshhold:
            return True
        else:
            return False
```

ДОДАТОК З.
ЛІСТИНГ КОДУ ДЛЯ СЕРВІСУ, ЩО ІМІТУЄ ПОВЕДІНКУ АПАРАТНОГО ВУЗЛА

```
import time
import uuid
import librosa
from models.Event import Event
from models.EventType import EventType
from models.CommandType import CommandType
import threading

class MockNode:
    def __init__(self, connector, settings, audioPath):
        self.__microEnabled = False
        self.__connector = connector
        self.__settings = settings
        self.__audioPath = audioPath
        self.id = str(uuid.uuid4())

    def start(self):
        connectEvent = Event(self.id, EventType.NodeConnected, None)
        self.__connector.receive(connectEvent)
        self.__enableMicro()

    def stop(self):
        disconnectEvent = Event(self.id, EventType.NodeDisconnected, None)
        self.__connector.receive(disconnectEvent)

        self.__microEnabled = False
```

```
    def receive(self, command):
        if command.type == CommandType.RecordAudio:
            self.__enableMicro()

    def __enableMicro(self):
        if not self.__microEnabled:
            self.__microEnabled = True
            threading.Thread(target=self.__recordAudioRoutine).start()

    def __recordAudioRoutine(self):
        recAudio, _ = librosa.load(self.__audioPath, sr=self.__settings.sampleRate)
        index = 0
        while self.__microEnabled:
            time.sleep(1)
            audioEvent = Event(self.id, EventType.AudioData, recAudio[index:index +
1000])
            self.__connector.receive(audioEvent)
            index += 10
            if index > len(recAudio) - 1:
                index = 0
```

ДОДАТОК К.
ЛІСТИНГ ПРОГРАМИ ЦЕНТРАЛЬНОГО МОДУЛЯ

```
import time
from models.Mode import Mode
from recognizers.AmbulanceRecognizer import AmbulanceRecognizer
from models.Command import Command
from models.EventType import EventType
from models.CommandType import CommandType
from models.Settings import Settings
from models.ConnectedNode import ConnectedNode
from connectors.ConnectorMock import NodeConnectorMock as NodeConnector
from loggers.LoggerConsole import LoggerConsole as Logger
import threading

class Main:
    def __init__(self):
        self.__settings = Settings(10, 41000)
        self.__connectedNodes = {}
        self.__ambulanceRecognizer = AmbulanceRecognizer(self.__settings)
        self.__connector = NodeConnector(self.__settings)
        self.__mode = Mode.AudioRecording
        self.__run()
        self.__logger = Logger()

    def __run(self):
        self.__connector.subscribe(self.__onDataReceived)
        self.__connector.start()

    def __onDataReceived(self, event):
```

```
        if event.type == EventType.NodeConnected:
            self.__onNodeConnected(event)
        elif event.type == EventType.NodeDisconnected:
            self.__onNodeDisconnected(event)
        elif event.type == EventType.AudioData:
            self.__onAudioDataReceived(event)
        elif event.type == EventType.VideoData:
            self.__onVideoDataReceived(event)

    def __onNodeConnected(self, event):
        if event.senderId not in self.__connectedNodes:
            self.__connectedNodes[event.senderId] = ConnectedNode(self.__settings,
event.senderId)

    def __onNodeDisconnected(self, event):
        if event.senderId in self.__connectedNodes:
            del self.__connectedNodes[event.senderId]

    def __onAudioDataReceived(self, event):
        if event.senderId not in self.__connectedNodes or self.__mode !=
Mode.AudioRecording:
            return
        senderNode = self.__connectedNodes[event.senderId]

        senderNode.audioBuffer.append(event.data)
        self.__logger.write(f"Audio buffer duration:
{senderNode.audioBuffer.duration()}")
        if senderNode.audioBuffer.duration() < 2:
            return
```

```
        audioData = senderNode.audioBuffer.getData()
        recognized = self.__ambulanceRecognizer.recognizeAudio(audioData)
        if not recognized:
            return

        self.__mode = Mode.WaitingToReset
        for node in self.__connectedNodes:
            if node.id == senderNode.id:
                command = Command(node.id, CommandType.EnableGreen)
            else:
                command = Command(node.id, CommandType.EnableRed)
            self.__connector.write(command)

        senderNode.audioBuffer.clear()
        threading.Thread(target=self.__waitToReset).start()

    def __waitToReset(self):
        time.sleep(30)

        if self.__mode != Mode.WaitingToReset:
            return

        for node in self.__connectedNodes:
            command = Command(node.id, CommandType.Reset)
            self.__connector.write(command)
            self.__mode = Mode.AudioRecording
Main()
```

ДОДАТОК Л.
ЛІСТИНГ КОДУ ДЛЯ ТЕСТУВАННЯ АУДІО-БУФФЕРА

```
import unittest
import numpy as np
from buffers.AudioBuffer import AudioBuffer

class AudioBufferTests(unittest.TestCase):
    def test_append(self):
        #Arrange
        buffer = AudioBuffer(1000)

        #Act
        buffer.append(np.zeros(1000))
        data = buffer.getData()

        #Assert
        self.assertEqual(len(data), 1000)

    def test_duration(self):
        #Arrange
        buffer = AudioBuffer(1000)

        #Act
        buffer.append(np.zeros(1000))
        duration = buffer.duration()

        #Assert
        self.assertEqual(duration, 1)

    def test_length(self):
```

```
        #Arrange
        buffer = AudioBuffer(1000)

        #Act
        buffer.append(np.zeros(1000))
        length = buffer.length()

        #Assert
        self.assertEqual(length, 1000)

    def test_clear(self):
        # Arrange
        buffer = AudioBuffer(1000)

        # Act
        buffer.append(np.zeros(1000))
        buffer.clear()
        data = buffer.getData()

        # Assert
        self.assertEqual(len(data), 0)


if __name__ == '__main__':
    unittest.main()
```

ДОДАТОК М.
ЛІСТИНГ СЕРВІСУ ТЕСТУВАННЯ

```
from connectors.ConnectorBase import NodeConnectorBase
from loggers.LoggerConsole import LoggerConsole as Logger
from mock.MockNode import MockNode
import uuid

class NodeConnectorMock(NodeConnectorBase):
    def __init__(self, settings):
        self.__logger = Logger()
        self.__subscribers = {}

        self.__nodes = [
            MockNode(self, settings, "static/ambulance_real.wav"),
            MockNode(self, settings, "static/alarm.wav")
        ]

    def start(self):
        self.__logger.write("Start connection.")
        for node in self.__nodes:
            node.start()

    def stop(self):
        self.__logger.write("Stop connection.")
        for node in self.__nodes:
            node.stop()

    def subscribe(self, callback):
        subid = str(uuid.uuid4())
        self.__subscribers[subid] = callback
```

```
def unsubscribe(self, subid):
    if subid in self.__subscribers:
        del self.__subscribers[subid]

def write(self, command):
    self.__logger.write(f"Sending: {command.type} to {command.receiverId}")
    for node in self.__nodes:
        node.receive(command)

def receive(self, event):
    self.__logger.write(f"Received: {event.type} from {event.senderId}")
    for sub in self.__subscribers.values():
        sub(event)
```

ДОДАТОК Н.
ДІАГРАМА ПОТОКУ ДАНИХ

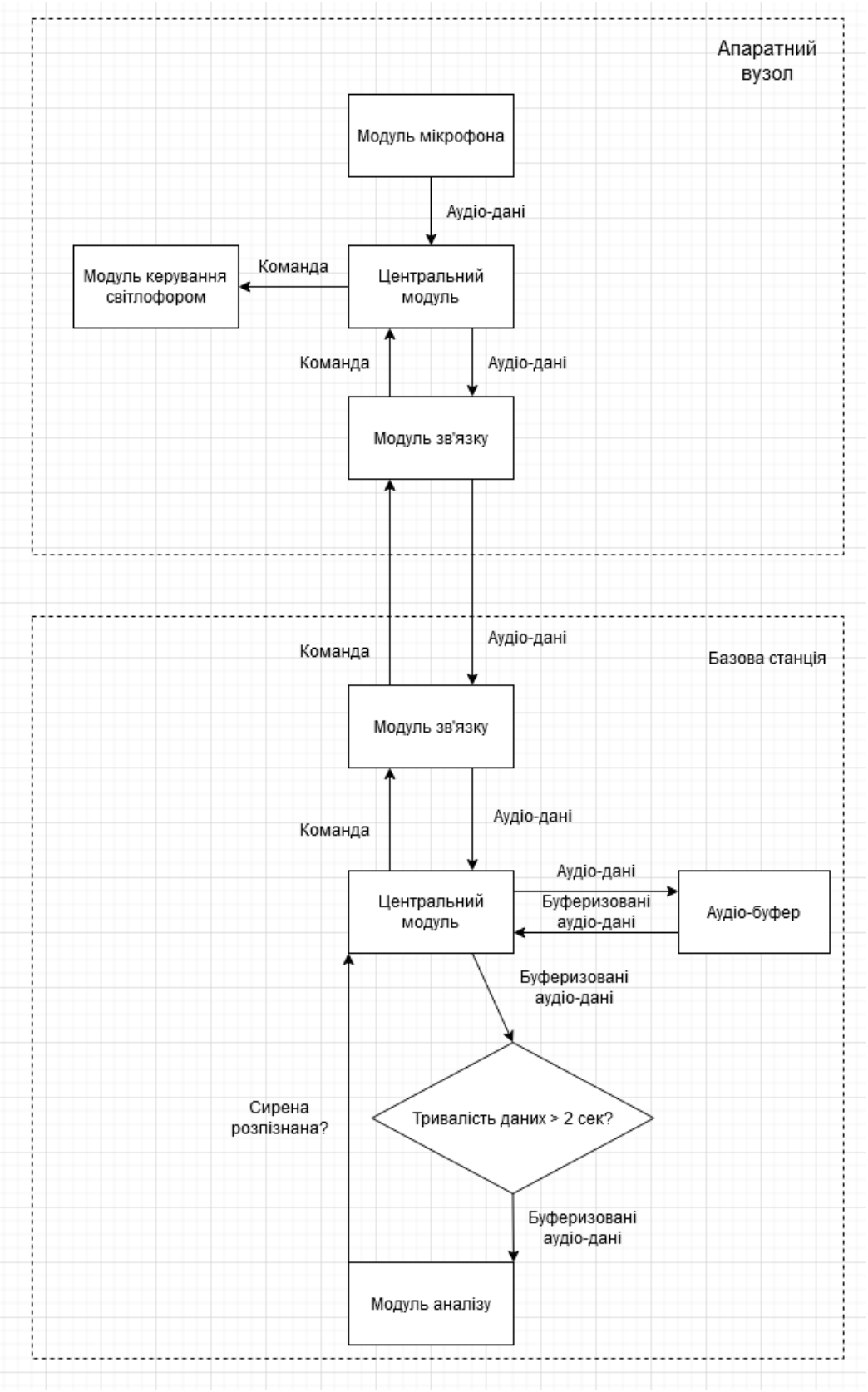


Рисунок Н.1. Діаграма потоку даних

ДОДАТОК П.
ЛІСТИНГ ПРОГРАМИ ДЛЯ ВІДОБРАЖЕННЯ СПЕКТРОГРАМИ АУДІО-СИГНАЛУ

```
import librosa.display
import matplotlib.pyplot as plt
import numpy as np

refAudioPath = "ambulance_ref.wav"
refAudioData, sampleRate = librosa.load(refAudioPath, sr=8000)
refAudioStft = abs(librosa.stft(refAudioData))

nsamples = len(refAudioData)

plt.plot(np.linspace(start=0, stop=nsamples / sampleRate, num=nsamples),
refAudioData)
plt.show()

plt.figure(figsize=(14, 14))
librosa.display.specshow(refAudioStft, sr=sampleRate, x_axis='time', y_axis='hz')
plt.colorbar()
plt.show()
```

ДОДАТОК Р.

ЛІСТИНГ ПРОГРАМИ ДЛЯ ВІДОБРАЖЕННЯ ПЕРЕТВОРЕННЯ ФУР'Є

```
import numpy as np
from scipy.fft import rfft, rfftfreq
import matplotlib.pyplot as plt
import librosa.display

refAudioPath = "ambulance_ref.wav"
refAudio, sampleRate = librosa.load(refAudioPath, sr=4000)

yf = rfft(refAudio)
xf = rfftfreq(len(refAudio), 1 / sampleRate)

plt.plot(xf, np.abs(yf))
plt.show()
```