МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ
Доктор техн. наук,
професор,
_____ І. М. Журавська
« __ » _____ 2023 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

СИСТЕМА ВИЯВЛЕННЯ ДЖЕРЕЛ ЗВУКУ НА БАЗІ MESH-МЕРЕЖІ

Спеціальність 123 Комп'ютерна інженерія
123 – КМР.1 – 605м.21710519

Студент _____К. А. Плюснін
підпис
«__»_____2023 р.

Керівник канд. фіз.–мат. наук, доцент _____С. В. Пузирьов
підпис
«__»_____2023 р.

Миколаїв – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CMOS	–	Complementary metal-oxide semiconductor
DAS	–	Delay-and-sum beamformer
DOA	–	Direction of arrival
FAS	–	Filter-and-sum beamformer
GPIO	–	General-purpose input/output
HTTP	–	HyperText Transfer Protocol
IDE	–	Integrated development environment
IoT	–	Interner of things
NAS	–	Network attached storage
OFDMA	–	Orthogonal frequency-division multiple access
PCB	–	Printed circuit board
SDRAM	–	Synchronous dynamic random access memory
SPIFFS	–	Serial peripheral interface flash file system
SRP	–	Steered response power
SSL	–	Sound source localization
TCP	–	Transmission Control Protocol
TDOA	–	Time difference of arrival
WSS	–	WebSocker secure

ВСТУП

Виявлення джерел звуку є актуальною проблемою як у цивільній, так і у військовій сферах.

У цивільних застосуваннях детектування звуку є дуже важливим компонентом пошуку людей під завалами, у печерах, горах та лісах, що широко використовується службами надзвичайних ситуацій.

У військовій сфері визначення напряму та відстані до джерела звуку є основним елементом контрбатарейної боротьби та протиповітряної оборони ближнього радіусу дії. Також системи детектування звуку можуть бути використанні в різноманітних охоронних системах.

Основною проблемою такого роду систем є чутливість до природних та штучних шумів, що знижує точність та якість визначення координат та напрямку джерела звуку. Також існуючі системи детектування джерел звуку не дозволяють точно визначити положення тих об’єктів, які генерують короткочасні звукові сигнали. Окрім того більшість систем акустичної локації не мають зручних інтерфейсів інтеграції для цивільних мереж зв’язку.

Актуальність кваліфікаційної роботи полягає у розробці охоронної системи на загальнодоступних апаратних компонентах з інтеграцією з мобільними мережами зв’язку, що дозволить оперативно інформувати користувачів про положення квазістатичних та короткочасних джерел звуку фактично в режимі реального часу.

Мета: дослідження теоретичних та методичних засад і розробка практичних рішень, що стосуються виявлення джерела звуку.

Об’єкт: Методи і алгоритми виявлення джерел звуку.

Предмет: Система охорони приміщення за допомогою виявлення джерел звуку з використанням mesh мережі на базі ESP32.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– аналіз інформації стосовно принципів та методів пошуку джерела звуку;

– порівняльний аналіз існуючих технічних рішень;

– обґрунтування вибору необхідних для реалізації системи компонентів;

– створення блок-схеми алгоритму акустичного комплексу;

– проектування пристрою на основі ESP32 та акустичних датчиків HC-SR04;

– створення концептуальної схеми системи визначення джерела звуку;

– створення діаграми послідовності системи визначення джерела звуку;

– створення діаграми апаратної архітектури системи визначення джерела звуку;

– створення діаграми програмної архітектури системи визначення джерела звуку.

– реалізація програмної частини тестового модулю апаратно-програмного комплексу.

Наукова новизна полягає у підвищенні точності визначення джерела звуку з використанням mesh-мережі акустичних датчиків.

Практичне значення отриманих результатів: результати дослідження та розроблення даної системи дозволять в режимі реального часу детектувати джерела звуку та інформувати користувачів про них.

Робота пройшла апробацію під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

Публікації. Основні положення та результати магістерської роботи опубліковані у збірнику матеріалів XXV Всеукраїнської науково-практичної конференції «Могилянські читання–2022» [30].

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ СИСТЕМИ, ЩО РОЗРОБЛЯЄТЬСЯ. ФОРМУВАННЯ ВИМОГ ДО АПАРАТНО-ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Порівняльний аналіз відомих технічних та програмних рішень

На даний момент існує безліч технічних рішень за темою локалізації джерел звуку, кожне з яких являється унікальним за метою, принципом роботи та реалізацією. Основна ідея пристроїв із локалізацією джерела звуку спрямована на визначення місця розташування джерела звуку з використанням різної кількості деяких датчиків або мікрофонів. Різняться вони між собою за комплексністю структури, точністю виявлення джерела звуку, надійністю та якістю.

1.1.1 Патент US10880669B2. Бінауральна локалізація джерела звуку

Слухач носить персональний пристрій для передачі звуку таким чином, що лівий навушник розташовується на лівому вусі, а правий навушник - на правому вусі. Для просторової локалізації звуку від джерела звуку персональний пристрій доставки звуку може мати масив з двох, трьох або більше мікрофонів, пов'язаних з кожним вухом, які спрямовані в різні боки для визначення місцезнаходження джерела звуку. Потім персональний пристрій доставки звуку виконує формування променя на виявленому звуці для визначення місцезнаходження джерела звуку. Функція моделювання сприйняття звуку, пов'язана з розташуванням джерела звуку, полегшує генерування звукових сигналів для просторової локалізації звуку від джерела звуку. Функція моделювання сприйняття звуку характеризує, як звук, що виводиться джерелом звуку, буде сприйматися слуховою системою людини на основі взаємодії з вухом, головою та/або тулубом. Персональний пристрій доставки звуку використовує цю функцію для штучного генерування звукових сигналів, які дозволяють слухачеві сприймати азимут, висоту, відстань звуку, що виводиться джерелом звуку. Потім персональний пристрій

доставки звуку виводить аудіосигнали для просторової орієнтації звуку для слухача на основі розташування джерела звуку.

Бінауральна локалізація джерела звуку спрямована на визначення місця розташування джерела звуку з використанням двох мікрофонів персонального пристрою доставки звуку, по одному пов'язаному з кожним вухом, а не двох, трьох або більше мікрофонів, пов'язаних з кожним вухом, які використовуються для виконання формування променя. Цими двома мікрофонами є лівий мікрофон, встановлений на лівому навушнику персонального пристрою доставки аудіосигналу, і правий мікрофон, встановлений на правому навушнику персонального пристрою доставки аудіосигналу. Визначення місцезнаходження джерела звуку за допомогою одного мікрофона, а не трьох мікрофонів для кожного вуха, зменшує складність, пов'язану з просторовою локалізацією звуку від джерела звуку [1].

1.1.2 Патент US10966022B1. Локалізація джерела звуку за допомогою декількох мікрофонних систем

У даному патенті описуються пристрої, що включають дві або більше мікрофонних систем різного розміру. Кожна мікрофонна система містить безліч мікрофонів, сконфігурованих для генерування сигнальнів у відповідь на звук. Мікрофони можуть бути розподілені в регулярному або нерегулярному лінійному, плоскому або тривимірному розташуванні. Також описані способи використання даних від мікрофонів у цих масивах для локалізації джерела звуку. Один з двох або більше масивів може бути обраний на основі заданого порогу. У деяких варіантах реалізації цей поріг може включати відстань. Наприклад, джерела звуку, відстань до яких перевищує порогове значення, локалізуються з використанням даних з великого масиву, а джерела звуку, відстань до яких не перевищує порогового значення, локалізуються з використанням даних з малого масиву.

Порогове значення може бути визначене шляхом порівняння результатів локалізації на основі даних з різних масивів з інформацією про місцезнаходження, зібраною за допомогою інших датчиків. Потім система може бути налаштована на використання масиву, який забезпечує більш точні результати для певного набору умов, таких як відстань і пеленг на джерело звуку. Наприклад, просторове положення джерела звуку може бути визначене у фізичному середовищі за допомогою різних методів, включаючи структуроване світло, захоплення зображення, ручне введення тощо. Структуроване світло може включати проекцію шаблону на об'єкти в межах сцени і може визначати положення на основі виявлення взаємодії об'єктів з шаблоном за допомогою пристрою для отримання зображень. Структура може бути регулярною, випадковою, псевдовипадковою тощо. Наприклад, система структурованого освітлення може визначати, що обличчя користувача знаходиться в певних координатах у приміщенні [2].

1.2 Огляд теоретичних відомостей стосовно принципів виявлення джерела звуку

Локалізація звуку полягає в знаходженні джерела звуку по відношенню до мережі мікрофонів. Здатність розрізняти та ідентифікувати конкретні звуки з навколишнього шуму є важливим аспектом нормальної слухової системи. Люди зі зниженим слухом страждають від того, що не можуть інтерпретувати мову в присутності фонового шуму і не можуть розпізнавати і розрізняти декількох мовців. Тому цей механізм реалізується в слухових апаратах для людей, які страждають від втрати слуху на одне або обидва вуха.

Протягом останніх двох-трьох десятиліть локалізація джерела звуку за допомогою набору мікрофонних систем була основною темою, що цікавила дослідників і обговорювалася в ряді заслуговуючих на увагу досліджень. До сьогоднішнього дня цій проблемі приділяється величезна увага з боку дослідників з області медицини, робототехніки та обробки сигналів. Однією з

багатьох проблем, що виникають у цій галузі, є проблема акустичної локалізації у реверберaційних середовищах. Крім того, кількість мікрофонів в установці та їх геометрія також є предметом постійних досліджень, оскільки існує необхідність обмежити використання мікрофонів в установці для того, щоб зробити систему компактною, зменшити складність та мінімізувати споживання ресурсів. Локалізація джерела звуку за допомогою мікрофонів все ще залишається теоретичною концепцією, яка продовжує активно досліджуватися.

1.2.1 Локалізація джерел звуку

Локалізація джерел звуку (SSL) – це широка сфера з багатьма важливими сферами застосування. У зовнішньому середовищі системи локалізації джерел звуку можуть, серед іншого, використовуватися для підвищення безпеки громадян шляхом виявлення та локалізації аномальних звуків, таких як постріли або крики. Локалізація є складним завданням, оскільки у випадку зовнішнього середовища на звуковий сигнал впливають погодні умови, шуми та об'єкти, що перекривають шлях розповсюдження. Крім того, все ще залишаються проблеми, пов'язані з реалізацією економічно ефективних алгоритмів, надійністю, точністю та балансуванням компромісів. SSL є сферою інтенсивних досліджень, і постійно публікуються нові роботи.

У повітрі звук створюється вібруючими частинками, які поширюються назовні в сферичному русі, і може бути описаний його характеристиками: довжиною хвилі, частотою, амплітудою і фазою. Таким чином, звук – це, по суті, хвилі, що поширюються в деякому середовищі. Як правило, можна вважати відносну початкову точку форми хвилі фазою, на рисунку 1.1 показано приклад зсуву фази між двома звуковими сигналами. Характеристики звукової хвилі, проілюстровані простою синусоїдою. Хвиля намальована пунктирною лінією, представляє другий звуковий сигнал і ілюструє приклад фазового зсуву фазового зсуву, також відомого як різниця фаз, між двома звуковими сигналами. Відстань, на яку поширюється

звуковий сигнал за період визначає довжину хвилі сигналу, а частота вимірюється в періодах в секунду. Отже, період T і довжину хвилі λ можна записати як $\lambda = Tc$, де c – швидкість звуку, а частота f може бути визначена як $f = c/\lambda$ [3].

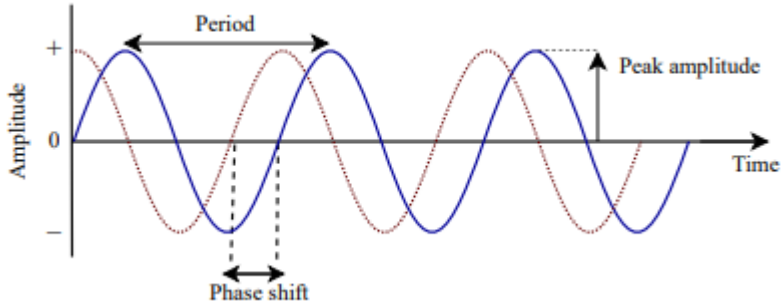


Рисунок 1.1 – Зсув фази між двома звуковими сигналами

На швидкість звуку можуть впливати різні змінні, такі як температура, тиск і густина. Оскільки температура зазвичай впливає на швидкість звуку найбільше, тиск і густина часто вважаються константами. Швидкість звуку в повітрі c може бути розрахована за допомогою рівняння (1.1).

$$c = 331.5\sqrt{1 + \frac{t}{273.15}} \tag{1.1}$$

де t - температура в градусах Цельсія.

Звукові сигнали часто представляються складними хвилями, які, якщо хвиля періодична, можна розкласти на більш прості синусоїдальні складові. Для отримання характеристик синусоїдальних складових, що складають складну хвилю, може бути використана математична формула, яка називається перетворенням Фур'є. Деталі різних варіантів та алгоритмів перетворення Фур'є виходять за рамки цієї роботи, але основна ідея полягає у відображенні сигналу з часової області в частотну. Отримані характеристики, які також називаються функціями, потім можуть бути використані для обчислення вимірювань, необхідних для оцінки положення.

1.2.2 Системи локалізації джерел звуку

Системи SSL залежать від здатності вловлювати та реєструвати звукові сигнали, що зазвичай досягається за допомогою акустичних датчиків. Для того, щоб зробити можливим запис отриманого звуку, датчики перетворюють звукові сигнали в електричну енергію. Датчики можна класифікувати за напрямками, з яких вони вловлюють звуки, загалом визначені як всеспрямовані, двонаправлені та односпрямовані. Всеспрямовані датчики виявляють звуки в усіх напрямках, тоді як двонаправлений виявляє звуки тільки з напрямку, на який спрямований датчик, і з його протилежного боку. Нарешті, односпрямований датчик вловлює звук тільки з того напрямку, куди він спрямований. Іншою технікою, що дозволяє досягти виявлення звуку в повній сфері, є амбісоніка. Мікрофон звукового поля, який складається з чотирьох односпрямованих мікрофонів, розташованих в тетраедрі, використовується для виявлення звукових сигналів і вироблення амбісонних форматів сигналів, відомих як A-формат і B-формат. Вироблені формати сигналів потім можуть бути використані для безпосереднього отримання напрямку джерела звуку [4].

На продуктивність систем локалізації звуку впливають тип використовуваних датчиків, а також їх кількість та розміщення у просторі. Може бути реалізовано кілька різних схем масивів з використанням різної кількості датчиків і різного їх розміщення, наприклад, бінауральна схема, три- і тетрауральна схема, кластерні масиви і некоординатні структури масивів. У три- та тетрауральних системах використовуються три та чотири мікрофони відповідно. Деякими поширеними типами датчиків є конденсаторні датчики, які перетворюють звукові сигнали в електричну напругу, датчики мікроелектромеханічних систем, які є невеликими і недорогими, і оптичні мікрофони, які дуже добре функціонують в суворих умовах.

Бездротові акустичні сенсорні мережі утворюються розподіленою групою акустико-сенсорних пристроїв, що мають можливості відтворення та запису звуку. Сучасні мобільні обчислювальні платформи надають широкі можливості для розробки додатків, пов'язаних з аудіо, із залученням акустико-сенсорних вузлів. У цьому контексті локалізація акустичних джерел є однією з прикладних областей, що привертає найбільшу увагу дослідницької спільноти протягом останніх десятиліть. У загальних рисах, локалізація акустичних джерел може бути досягнута шляхом вивчення енергетичних та часових і/або спрямованих характеристик вхідного звуку на різних мікрофонах та використання відповідної моделі, яка пов'язує ці характеристики з просторовим розташуванням джерела (або джерел), що представляє інтерес.

Коли вузли включають в себе акустичні перетворювачі, а обробка включає в себе маніпуляції з аудіосигналами, результуюча мережа зазвичай називається бездротовою акустичною сенсорною мережею.

Існує два типових завдання локалізації в бездротових акустичних сенсорних мережах: локалізація одного або декількох джерел звуку, що представляють інтерес, і локалізація вузлів, що складають мережу. Перший випадок може бути пов'язаний з визначенням місцезнаходження невідомих джерел звуку, наприклад, гучномовців в приміщенні або несподіваних акустичних подій, а іноді й інших пристроїв, що випромінюють відомі радіомаякові сигнали. Другий випадок зазвичай пов'язаний з самокалібруванням або автоматичним визначенням дальності самих вузлів.

1.3 Методи локалізації джерел звуку

Для оцінки місцезнаходження джерел звуку, які є активними в акустичному середовищі, що контролюються бездротовими акустичними мережами, існують різні методи. Зазвичай приймається централізована схема, коли виділений вузол, відомий як центр злиття, відповідає за виконання завдання локалізації на основі інформації, яку він отримує від

решти вузлів. Сама сенсорна мережа створює багато обмежень і проблем, які необхідно враховувати при розробці підходу до локалізації, щоб полегшити її використання в практичних сценаріях. Такі проблеми включають використання пропускної здатності, що обмежує кількість інформації, яка може бути передана в мережі, та обмежену обчислювальну потужність вузлів, що не дозволяє їм виконувати дуже складні та обчислювально дорогі операції. Крім того, кожен вузол має власний тактовий генератор для дискретизації сигналів, а оскільки вузли працюють індивідуально, то результуючий звук в мережі не буде синхронізованим.

Методи SSL можуть бути побудовані на основі характеру інформації від датчиків, яка використовується для оцінки місцезнаходження. Таким чином, бездротова акустична мережа може оцінювати місцезнаходження акустичних джерел на основі показань енергії вхідних сигналів, їх часу надходження або різниці в часі надходження, напрямку надходження і потужності керованого відгуку, що виникає в результаті об'єднання декількох мікрофонних сигналів.

1.3.1 Метод напрямку надходження (DOA)

У підходах, що базуються на DOA, кожен вузол оцінює напрямок надходження джерел, які він може виявити, і передає відповідні оцінки в центр синтезу. Хоча такі підходи вимагають підвищеної обчислювальної потужності і декількох датчиків в кожному вузлі, вони можуть досягти дуже низького використання пропускної здатності, оскільки потрібно передавати тільки оцінки DOA. Крім того, оскільки оцінка здійснюється в кожному вузлі окремо, аудіосигнали на різних вузлах не повинні бути синхронізовані. Підходи, засновані на DOA, можуть терпіти несинхронізовані вхідні дані до тих пір, поки джерела рухаються з досить повільною швидкістю відносно кадру аналізу. Виявлення місцезнаходження, як правило, базуються на оцінці місцезнаходження джерела звуку як перетину ліній, що виходять з вузлів. Однак для декількох джерел виникає кілька проблем: кількість виявлених

джерел (і, отже, кількість оцінок DOA) в кожен момент часу може змінюватися по вузлах через пропущені виявлення (тобто джерело не виявлено вузлом) або хибні тривоги (тобто переоцінка кількості виявлених джерел), і необхідна процедура об'єднання для пошуку комбінацій напрямків надходження, які відповідають одному джерелу. Це називається проблемою асоціації даних і має вирішальне значення для задачі локалізації.

При оцінці DOA джерела звуку зазвичай використовується модель розповсюдження в дальньому полі. Модель далекого поля може бути описана як відстань між датчиками в масиві, що є досить малою по відношенню до відстані до джерела звуку, що дозволяє розглядати звукову хвилю як плоску, а не сферичну. У двовимірному випадку напрямок надходження можна виразити за допомогою полярних координат і чотириквадрантної функції оберненої дотичної, як показано в рівнянні (1.2).

$$\theta = \arctan\left(\frac{y_s - y_i}{x_s - x_i}\right) \tag{1.2}$$

де (x_s, y_s) – місцезнаходження джерела, а $(x_i\ , y_i)$ – місце розташування i-го датчика. Для тривимірного виявлення джерела звуку потрібні оцінки як азимуту θ, так і висоти ϕ. Азимут вже визначено рівнянням (1.2), а висота може бути сформульована за допомогою рівняння (1.3) [5].

$$\theta = \arctan\left(\frac{z_s - z_i}{\sqrt{(x_s - x_i)^2 + y_s - y_i{}^2}}\right) \tag{1.3}$$

1.3.2 Метод тріангуляції

Основна теорія тріангуляції полягає в тому, що можна оцінити положення точки всередині трикутника точки всередині трикутника, припускаючи, що розташування вершин трикутника відоме, а також кути, під якими точка бачить вершини. Таким чином, для локалізації одного джерела кожне вимірювання DOA генерує лінію, яка починається в місці розташування датчика і продовжується в заданому напрямку. Положення джерела потім оцінюється як перетин двох або більше таких ліній. У реальних сценаріях прийнятий звуковий сигнал, найчастіше, забруднений

шумом, і в результаті метод тріангуляції не зможе дати однозначного рішення. Одним з підходів до вирішення вищезазначеної проблеми є формулювання максимальної оцінки ймовірності та застосування зваженої нелінійної функції вартості за методом найменших квадратів для отримання оцінки положення.

Через нелінійність функції витрат необхідно прийняти ітераційну функцію мінімізації, таку як симплекс Ньютона-Рафсона, Гаусса-Ньютона або Нелдера-Міда. Також були запропоновані інші підходи, які натомість спрямовані на лінеаризацію функції витрат для отримання розв'язку в замкненій формі. Відомим прикладом такого підходу є алгоритм Стенсфілда. Припускаючи, що виникатимуть лише невеликі помилки максимальної оцінки ймовірності, Стенсфілд запропонував замінити залишок від функції витрат синусом залишку, що призведе до лінійної функції найменших квадратів. Недоліком є те, що припущення про малу похибку часто негативно впливає на точність алгоритму позиціонування через збільшене зміщення.

1.3.3 Метод вимірювання різниці часу прибуття (TDOA)

TDOA пов'язано з різницею в часі польоту хвильового потоку, що створюється джерелом в парі датчиків в одному і тому ж вузлі. Його можна оцінити з помірними обчислювальними витратами за допомогою узагальненої перехресної кореляції сигналів, отриманих датчиками в парі. Виявлення місцезнаходження джерела виконується шляхом об'єднання вимірювань різниць часу прибуття, що надходять від декількох датчиків. Хоча TDOA і підходить для бездротових сенсорних мереж, в практичних сценаріях (реверберaційні середовища, наявність шумів і завад), такі вимірювання схильні до помилок, які, в свою чергу, призводять до неправильної локалізації. Для того, щоб зменшити вплив цих негативних явищ, було запропоновано декілька методик з метою виявлення та усунення викидів у наборі TDOA [6].

Основна ідея вимірювання різниці часу прибуття полягає у використанні різниці в часі приходу сигналів між N парами сенсорних вузлів. Припускаючи, що сенсорні вузли мають декілька датчиків, обчислення можуть бути виконані окремо кожним вузлом. Центральному процесору тоді потрібно лише об'єднати отримані вимірювання для оцінки положення джерела, це робить систему менш чутливою до проблем синхронізації між вузлами. Оскільки звук рухається сферично, зі швидкістю звуку c, сигнал буде досягати мікрофонів у дещо різний час. Таким чином, час проходження сигналу, τ_{ij} , між i-м та j-м мікрофонами може бути сформульований за допомогою рівняння (1.4).

$$\tau_{ij} = \frac{d_{ij}}{c} = \frac{\|x_s - r_i\| - \|x_s - r_j\|}{c} \quad (1.4)$$

де d_{ij} – різниця відстаней від джерела до i-го та j-го мікрофонів, r_i та r_j – місцезнаходження i-го та j-го датчиків, x_s – місцезнаходження джерела.

1.3.4 Метод енергетичної локалізації

Енергетична локалізація ґрунтується на усереднених значеннях енергії, обчислених на інтервалах відліків сигналів, отриманих датчиками, встановленими у вузлах. У порівнянні з методами TDOA і DOA, енергетичні підходи привабливі тим, що вони не вимагають використання декількох датчиків у вузлах і вільні від проблем синхронізації, на відміну від підходів, заснованих на часу надходження. Однак, методи на основі TDOA і DOA, що розглядаються як сигнальні підходи, пропонують в цілому кращу продуктивність, ніж енергетичні методи. Це пов'язано з тим, що використовується безпосередньо інформація, яка передається всіма відліками сигналу, а не їх середнє значення, за рахунок більш складних пристроїв захоплення і ресурсів передачі.

Енергетична локалізація використовує акустичне затухання, тобто базується на тому, що енергія сигналу зменшується з відстанню. Інтенсивність звуку, або енергія, підпорядковується закону оберненого

квадрата, що означає, що спад енергії обернено пропорційний квадрату відстані від джерела звуку. Отримана акустична енергія, яку іноді називають функцією спаду енергії, в загальному випадку виражається рівнянням (1.5).

$$y(t) = g\frac{S(t)}{d^2(t)} + \epsilon(t) \tag{1.5}$$

де $y(t)$ – отримана енергія, g - коефіцієнт підсилення, $S(t)$ – енергія звуку в одному метрі від джерела звуку, $d(t)$ – відстань між датчиками та джерелом звуку $\epsilon(t)$ – адитивний шум.

Першим кроком у локалізації джерела звуку є отримання енергетичного співвідношення для кожної пари акустичних датчиків, що може бути зроблено шляхом використання енергії отриманого сигналу, де співвідношення, k_{ij} , між i-м та j-м датчиками може бути сформульовано рівнянням (1.6).

$$k_{ij} := \left(\frac{y_i/y_j}{g_i/g_j}\right)^{-\frac{1}{2}} = \frac{||x_s - r_i||}{||x_s - r_j||}, \tag{1.6}$$

де $k_{ij} \neq 1$, x_s - місцезнаходження джерела, r_i та r_j - місцезнаходження датчиків. Всі можливі x_s, що задовольняють рівнянню (1.3), розташовані на гіперсфері, яка описується рівнянням (1.7).

$$||x_s - c_{ij}||^2 = p_{ij}^2, \tag{1.7}$$

де c_{ij} - центр гіперсфери, а p_{ij} - радіус, які можна визначити за допомогою рівняннь (1.8), (1.9).

$$c_{ij} = \frac{r_i - {k_{ij}}^2 r_j}{1 - {k_{ij}}^2}, \tag{1.8}$$

$$p_{ij} = \frac{k_{ij}||r_i - r_j||}{1 - {k_{ij}}^2}, \tag{1.9}$$

1.3.5 Метод потужності керованої реакції (SRP)

Підходи з потужністю керованої реакції – це методи, засновані на формуванні променю, які обчислюють вихідну потужність фільтра і сумарного формувача променю, що спрямовується на множину потенційних

місць розташування джерела, визначених заздалегідь визначеною просторовою сіткою. Цей метод спрямований на максимізацію потужності прийнятого звукового сигналу за допомогою FAS або DAS формувача променя.

Оскільки обчислення SRP включає в себе накопичення узагальненої перехресної кореляції від декількох пар мікрофонів, вимоги до синхронізації, як правило, такі ж, як і в методах, заснованих на вимірюванні різниці часу прибуття. Сукупність SRP, отриманих в різних точках мережі, складають карту SRP, де точка, що накопичує найбільше значення, відповідає передбачуваному місцезнаходженню джерела. При використанні несинхронізованих вузлів з декількома мікрофонами, карти SRP, розраховані на кожному вузлі, можуть бути використані для отримання відповідних напрямків надходження. Альтернативно, карти SRP від різних датчиків можуть бути накопичені в центральному вузлі для отримання комбінованої карти SRP, що ідентифікує справжнє місце розташування джерела за його максимумом.

Алгоритм керованої потужності відгуку є широко використовуваним алгоритмом для виявлення джерела звуку на основі формування променя. Загальна ідея полягає в тому, щоб розбити область дослідження на сітку, що містить всі можливі місця розташування джерела. Потім проводиться пошук по сітці на основі карти потужності, отриманої за допомогою алгоритму формування променя, де передбачувані місця розташування джерела визначаються піками на карті потужності. Алгоритм керованої потужності відгуку зазвичай реалізується з використанням розподіленої обробки, де кожна мікрофонна решітка виробляє власну карту потужності, яка потім відправляється на центральний процесор для обчислення остаточної оцінки положення.

1.4 Огляд теоретичних відомостей стосовно поняття mesh-мереж

1.4.1 Визначення mesh-мережі

Mesh-мережа – це мережа, в якій вузли з'єднані між собою, відгалужуючись від інших пристроїв або вузлів, що налаштовані для ефективної маршрутизації даних між пристроями та клієнтами та допомагають організаціям забезпечити стабільне з'єднання у всьому фізичному просторі.

Топології комірчастих мереж створюють кілька маршрутів для переміщення інформації між підключеними вузлами. Такий підхід підвищує відмовостійкість мережі в разі відмови вузла або з'єднання. Великі комірчасті мережі можуть включати кілька маршрутизаторів, комутаторів та інших пристроїв, які працюють як вузли. Комірчаста мережа може включати сотні бездротових комірчастих вузлів, що дозволяє їй охоплювати велику територію.

Існує два типи mesh-мереж: повна комірчаста мережа та часткова комірчаста мережа. Як показано на рис. 1.2, повна комірчаста мережа – це мережа, в якій всі вузли можуть безпосередньо зв'язуватися з кожним іншим вузлом.

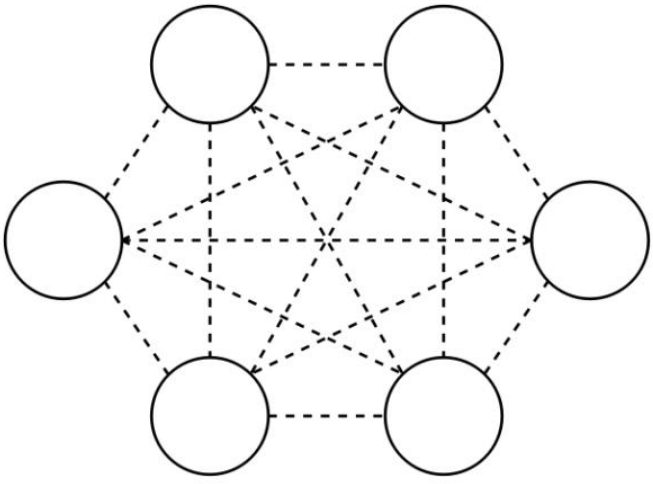


Рисунок 1.2 – Повна комірчаста мережа

У частковій комірчастій мережі вузли не можуть безпосередньо зв'язуватися один з одним. Але, як показано на рис. 1.3, всі вузли все ще можуть зв'язатися з кожним іншим вузлом мережі. Це досягається за допомогою одного або декількох проміжних вузлів, які ретранслюють зв'язок до вузла призначення.

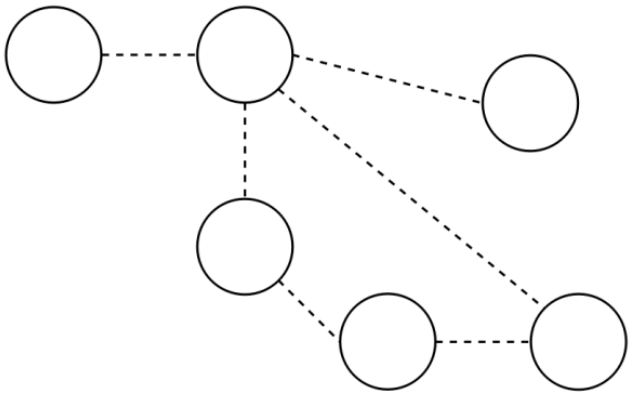


Рисунок 1.3 – Часткова комірчаста мережа

З'єднання в повній або частковій мережі можуть бути дротовими або бездротовими комірчастими мережами. Рішення про використання повної або часткової комірчастої мережі залежить від таких факторів, як загальна структура трафіку в мережі та ступінь ризику виходу з ладу вузлів або з'єднань [7].

1.4.2 Принцип роботи mesh-мережі

Бездротові mesh-мережі можуть легко, ефективно і бездротово з'єднувати великі території, використовуючи недорогу існуючу технологію. У бездротовій комірчастій мережі мережеве з'єднання розподіляється між десятками або навіть сотнями бездротових комірчастих вузлів, які комунікують один з одним, щоб розділити мережеве з'єднання на великій території.

Традиційний мережевий маршрутизатор слугує концентратором для пристроїв, підключених до його мережі. Більшість традиційних "бездротових" точок доступу все ще потребують дротового підключення до

інтернету для трансляції свого сигналу. Для великих бездротових мереж кабелі Ethernet потрібно прокладати в стелі, стінах і громадських місцях.

У бездротовій mesh-мережі лише один вузол повинен мати підключення до інтернету. Кожен вузол, доданий до мережі, ділиться своїм з'єднанням бездротовим способом з усіма іншими вузлами в його околицях, використовуючи один з декількох протоколів. Чим більше вузлів, тим далі поширюється з'єднання, створюючи бездротову "хмару зв'язку", яка може обслуговувати великий офіс або місто-мільйонник.

Якщо потрібно забезпечити доступом до Інтернету лише невелику територію, навряд чи можна побачити велику різницю між окремим вузлом mesh-мережі і традиційним бездротовим маршрутизатором. Однак для великих мереж mesh-мережі мають ряд переваг [8]:

– Використання меншої кількості дротів означає, що створення великої мережі коштує дешевше.

– Чим більше вузлів встановлюється, тим більшою і швидшою стає бездротова мережа.

– Вони покладаються на ті самі стандарти бездротового зв'язку, що й більшість бездротових мереж.

– Вони зручні там, де немає настінних Ethernet-з'єднань, зокрема на відкритих майданчиках і в місцях, де традиційна інфраструктура не працює.

– Вони корисні для мережевих конфігурацій за межами прямої видимості, де бездротові сигнали періодично блокуються.

– Mesh-мережі самоконфігуруються, мережа автоматично включає новий вузол в існуючу структуру, не потребуючи жодних налаштувань з боку мережевого адміністратора.

– Mesh-мережі самовідновлюються, оскільки мережа автоматично знаходить найшвидші та найнадійніші шляхи для передачі даних, навіть якщо вузли заблоковані або втрачають сигнал.

– Бездротові комірчасті конфігурації дозволяють локальним мережам працювати швидше, оскільки локальним пакетам не потрібно повертатися назад до центрального сервера.

– Бездротові mesh-вузли легко встановлювати та видаляти, що робить мережу надзвичайно адаптивною та розширюваною, коли потрібне більше або менше покриття.

1.5 Практичні вимоги до системи визначення положення джерела звуку

При проектуванні систем локалізації з використанням бездротових акустичних мереж виникають деякі реальні проблеми. Для побудови надійної і точної системи локалізації необхідно забезпечити компроміс між аспектами, пов'язаними з вартістю, енергоефективністю, простотою калібрування, складністю розгортання і точністю. Досягнення такого компромісу не є простим завданням і охоплює багато практичних проблем.

1.5.1 Економічна ефективність

Потенціал бездротових акустичних мереж щодо забезпечення високоточних функцій акустичної локалізації в значній мірі залежить від базових апаратних технологій у вузлах. Наприклад, методи локалізації, засновані на DOA, TDOA або SRP, потребують інтенсивної внутрішньовузлової обробки для обчислення узагальненої перехресної кореляції, а також вхідних ресурсів, що дозволяють здійснювати багатоканальний аудіозапис. З появою потужних одноплатних комп'ютерів можна легко досягти високопродуктивної внутрішньовузлової обробки сигналів. Тим не менш, слід враховувати важливі аспекти щодо вартості та розсіювання енергії, особливо для вузлів з батарейним живленням, які масово розгортаються.

1.5.2 Особливості впровадження

Методи виявлення джерела звуку зазвичай вимагають процесу конфігурації перед розгортанням. Наприклад, методи на основі часу надходження зазвичай потребують належного налаштування механізмів синхронізації перед початком локалізації цілей. Аналогічно, енергетичні методи потребують вузлів з відкаліброваними коефіцієнтами підсилення для отримання високоякісних вимірювань енергетичного співвідношення. Усі ці завдання зазвичай є складними і трудомісткими. Більше того, вони, як правило, потребують людського нагляду під час автономної фази профілювання. Така фаза перед розгортанням може стати ще складнішою в деяких прикладних середовищах, де доступ до вузлів може бути отриманий неавторизованими суб'єктами. Крім того, бездротові акустичні мережі також застосовуються за межами закритих будівель. При цьому вони схильні до добових і сезонних коливань температури і відповідних варіацій швидкості звуку. Щоб впоратися з цим недоліком, калібрування необхідно автоматизувати і зробити адаптивним до навколишнього середовища.

1.5.3 Відмовостійкість системи

Бездротова акустична мережа повинна також реалізовувати механізми самоконфігурації, що враховують динаміку мережі, наприклад, пов'язану з відмовами вузлів. У цьому контексті при проектуванні системи необхідно враховувати кількість якірних вузлів, необхідних для розгортання, та стратегію їх розміщення. Система повинна гарантувати, що якщо деякі з вузлів вийдуть з мережі, решта все одно зможуть надавати оцінки місцезнаходження належним чином. З цією метою важливо максимізувати зону покриття при мінімізації кількості необхідних опорних вузлів в системі.

1.5.4 Масштабованість системи

Залежно від конкретного застосування, бездротова акустична мережа, яку необхідно розгорнути, може варіюватися від дуже маленької і простої

мережі з декількох вузлів до дуже великих мереж з десятками або сотнями вузлів і складною топологією мережі. Наприклад, в додатках для моніторингу дикої природи використовується дуже велика кількість датчиків для акустичного моніторингу дуже великих середовищ, в той час як топологія мережі може постійно змінюватися через те, що датчики зміщуються вітром або тваринами, що проходять повз. Викликом для таких застосувань є розробка методів локалізації і самоконфігурації, які можуть бути легко масштабовані до складних бездротових акустичних мереж.

1.5.5 Похибки вимірювання

Радіочастотна локалізація в бездротових акустичних мережах схильна до помилок через нерегулярні схеми поширення, викликані умовами навколишнього середовища (тиск і температура) і випадковими багатопроменевими ефектами, такими як відбиття, заломлення, дифракція і розсіювання. У випадку бездротових акустичних мереж акустичні сигнали також піддаються подібним спотворенням, спричиненим змінами навколишнього середовища та ефектами, спричиненими шумом і джерелами завад, відбитими відлуннями, перешкодами від об'єктів або дифракцією сигналу. Інші помилки пов'язані з вищезгаданим процесом перед розгортанням, що призводить до помилок синхронізації або неточностей у положеннях якірних вузлів. Ці помилки повинні бути проаналізовані для того, щоб відфільтрувати шуми вимірювань і підвищити точність оцінок місцеположення.

1.5.6 Робота в режимі реального часу

Системи визначення положення джерела звуку вимагають роботи в режимі реального часу, коли результати вимірювань потрібно отримувати без затримки. Система повинна бути здатна до роботи в режимі реального часу, що дозволяє вчасно виявляти небезпечні ситуації або здійснювати контроль за технологічним процесом.

Висновки до розділу 1

В цьому розділі було проведено порівняльний аналіз існуючих технічних рішень. Наступним чином проведено огляд теоретичних відомостей стосовно принципів виявлення джерела звуку, розглянуто існуючі методи локалізації джерел звуку, теоретичні відомості стосовно поняття mesh-мереж, які можуть бути використані для побудови систем визначення положення джерела звуку, а також поставлено практичні вимоги до системи визначення положення джерела звуку.

2 ПІДБІР АПАРАТНО-ПРОГРАМНИХ КОМПОНЕНТІВ

2.1 Огляд компонентів апаратно-програмного комлексу

Перш ніж переходити до проектування апаратно-програмного комплексу необхідно провести аналіз доступних мікроконтролерів, датчиків, сенсорів та на основі зібраних даних обрати необхідні компоненти для реалізації системи визначення положення джерела звуку.

2.1.1 Вибір мікроконтролера системи

Arduino – це платформа для розробки мікроконтролерів в поєднанні з інтуїтивно зрозумілою мовою програмування, прошивку для якої розробляють за допомогою інтегрованого середовища розробки (IDE) Arduino. Основна перевага платформи Arduino полягає в тому, що за допомогою неї можна створити майже усе, що завгодно. Розглядуваний мікроконтролер можна оснащати різними датчиками, модулями, сенсорами та іншими додатковими компонентами, таким чином перетворивши Arduino в програмований «мозок» практично для будь-якої системи управління. Можливості платформи обмежені тільки людською уявою, а також апаратними обмеженнями, що можна вирішити за допомогою заміни поточного мікроконтролера на більш потужний.

Плата Arduino Nano (рис. 2.1) являється однією з найдешевших і найпоширеніших плат. Цей мікроконтролер – найкращий варіант для початку роботи з платформою через свій оптимальний розмір, популярність, а також наявність великої кількості безкоштовних уроків і прикладів програм.

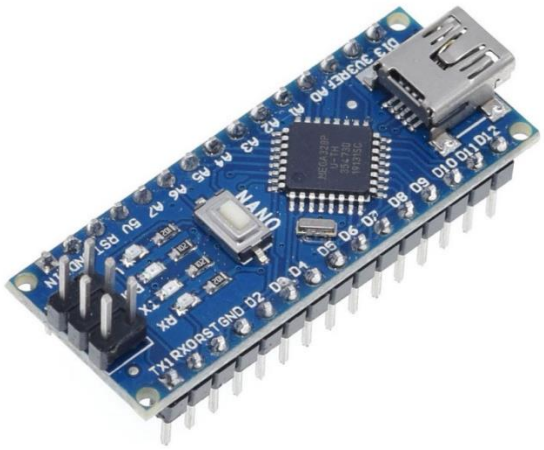


Рисунок 2.1 – Зовнішній вигляд плати Arduino Nano

Arduino Nano – це невелика, повна і зручна для макетування плата на базі ATmega328 (Arduino Nano 3.x). Вона має більш-менш таку ж функціональність, як і Arduino Duemilanove, але в іншому корпусі. У ній відсутній лише роз'єм живлення постійного струму, і вона працює з USB-кабелем Mini-B замість стандартного.

Arduino Nano має ряд можливостей для зв'язку з комп'ютером, іншим Arduino або іншими мікроконтролерами. ATmega328 забезпечує послідовний зв'язок UART TTL (5В), який доступний на цифрових виводах 0 (RX) і 1 (TX). FTDI FT232RL на платі передає цей послідовний зв'язок через USB, а драйвери FTDI (включені в програмне забезпечення Arduino) забезпечують віртуальний COM-порт для програмного забезпечення на комп'ютері. Програмне забезпечення Arduino включає в себе послідовний монітор, який дозволяє надсилати прості текстові дані на плату Arduino та з неї. Світлодіоди RX і TX на платі блимають, коли дані передаються через мікросхему FTDI і USB-з'єднання з комп'ютером (але не для послідовного зв'язку на виводах 0 і 1). Бібліотека SoftwareSerial дозволяє здійснювати послідовний зв'язок через будь-який з цифрових виводів Nano. ATmega328 також підтримує I2C (TWI) і SPI зв'язок. Програмне забезпечення Arduino включає бібліотеку Wire для спрощення використання шини I2C. Для

використання SPI-зв'язку, будь ласка, зверніться до специфікації ATmega328 [9]. Характеристики Raspberry Pi 4 B наведені у табл. 2.1 [10].

Таблиця 2.1 – Характеристики Arduino Nano

Мікроконтролер	ATmega328P
Тактова частота	16 МГц
Робоча напруга	5 В
Вхідна напруга	7-12 В
Сила струму на входах / виходах	40 мА
Сила струму для 3.3В виходу	50 мА
Пам'ять	32 кБ
SRAM	2 кБ
EEPROM	1 кБ
Цифрових входів / виходів	14
Аналогових входів	8

ESP32 (рис. 2.2) – багатофункціональний мікроконтролер з інтегрованим Wi-Fi та Bluetooth для широкого спектру додатків. Він розроблений для досягнення найкращих показників енергоспоживання та радіочастотної продуктивності, демонструючи міцність, універсальність та надійність у широкому спектрі застосувань та сценаріїв живлення.

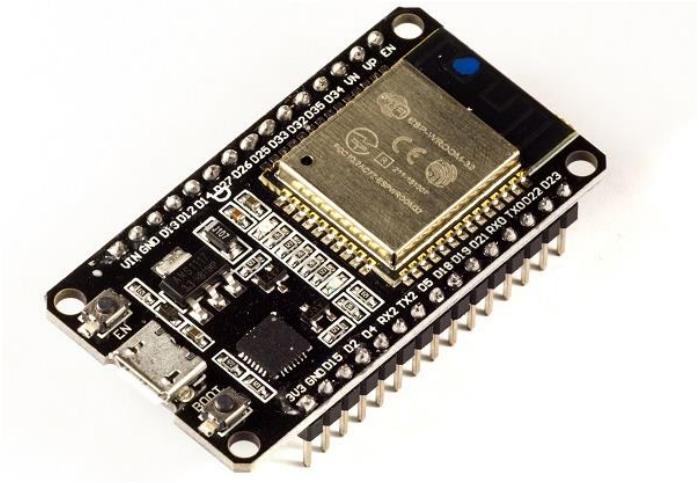


Рисунок 2.2 – Зовнішній вигляд мікроконтролеру ESP32

Мікроконтролер ESP32 розроблений для мобільної, натільної електроніки та додатків Інтернету речей (IoT). Він має всі найсучасніші характеристики малопотужних мікросхем, включаючи дрібнозернистий тактовий генератор, кілька режимів живлення та динамічне масштабування потужності. Наприклад, у сценарії малопотужного концентратора датчиків IoT ESP32 періодично прокидається тільки при виявленні певного стану. Низький робочий цикл використовується для мінімізації кількості енергії, яку витрачає мікросхема. Вихідна потужність підсилювача потужності також регулюється, що сприяє досягненню оптимальному співвідношенню між дальністю зв'язку, швидкістю передачі даних і енергоспоживанням.

ESP32 складається з антенного перемикача, радіочастотного балуна, підсилювача потужності, малошумного приймального підсилювача, фільтрів і модулів керування живленням. Таким чином, все рішення займає мінімальну площу друкованої плати (PCB). ESP32 використовує CMOS для однокристальної повністю інтегрованої радіо- та базової смуги, а також застосовує вдосконалені калібрувальні схеми, які дозволяють усунути зовнішні недоліки схеми або пристосуватися до змін зовнішніх умов. Таким чином, масове виробництво рішень на основі ESP32 не потребує дорогого та спеціалізованого обладнання для тестування Wi-Fi [11].

Особливості мікроконтролеру ESP32 [12]:

– Надійна конструкція. ESP32 здатний надійно функціонувати в промислових умовах в діапазоні робочих температур від -40°C до +125°C. Завдяки вдосконаленим калібрувальним схемам ESP32 може динамічно усувати зовнішні недоліки схеми та адаптуватися до змін зовнішніх умов.

– Наднизьке енергоспоживання. Розроблений для мобільних пристроїв, натільної електроніки та додатків Інтернету речей, ESP32 забезпечує наднизьке енергоспоживання завдяки поєднанню декількох типів фірмового програмного забезпечення. ESP32 також включає в себе

найсучасніші функції, такі як дрібнозернистий тактовий генератор, різні режими живлення та динамічне масштабування потужності.

– Високий рівень інтеграції. ESP32 має високу ступінь інтеграції з вбудованими антенними перемикачами, радіочастотним балуном, підсилювачем потужності, малошумним підсилювачем прийому, фільтрами та модулями керування живленням. ESP32 додає безцінну функціональність і універсальність вашим додаткам з мінімальними вимогами до друкованої плати (PCB).

– Гібридний чіп Wi-Fi та Bluetooth. ESP32 може працювати як повноцінна автономна система або як підлеглий пристрій до головного MCU, зменшуючи накладні витрати на комунікаційний стек основного процесора програми. ESP32 може взаємодіяти з іншими системами для забезпечення функціональності Wi-Fi і Bluetooth через інтерфейси SPI / SDIO або I2C / UART.

Характеристики ESP32 наведені у табл. 2.2 [13].

Таблиця 2.2 – Характеристики ESP32

Мікроконтролер	Tensilica Xtensa LX6
Тактова частота	240 МГц
Максимальний струм споживання	260 мА
SRAM	520 Кб
Стандарти бездротового зв'язку	Wi-Fi: 802.11 b / g / N, Bluetooth: v4.2 BR/EDR and BLE
Робоча температура	-40ºC to 125ºC

2.1.2 Вибір мікрокомп’ютеру для серверу бази даних

Raspberry Pi 4 Model B (рис. 2.3) – це найновіший продукт у популярній лінійці комп'ютерів Raspberry Pi. Він пропонує революційне збільшення швидкості процесора, мультимедійної продуктивності, пам'яті та можливостей підключення порівняно з попереднім поколінням Raspberry Pi 3 Model B+, зберігаючи при цьому зворотну сумісність та аналогічне енергоспоживання. Для кінцевого користувача Raspberry Pi 4 Model B забезпечує продуктивність настільного комп'ютера, яку можна порівняти з комп'ютерними системами x86 початкового рівня.

Ключові особливості цього продукту включають високопродуктивний 64-бітний чотириядерний процесор, підтримку двох дисплеїв з роздільною здатністю до 4K через пару портів micro-HDMI, апаратне декодування відео з роздільною здатністю до 4Kp60, 4 ГБ оперативної пам'яті, дводіапазонний діапазон 2.4/5.0 ГГц бездротова локальна мережа, Bluetooth 5.0, гігабітний Ethernet, USB 3.0 і підтримка PoE [14].

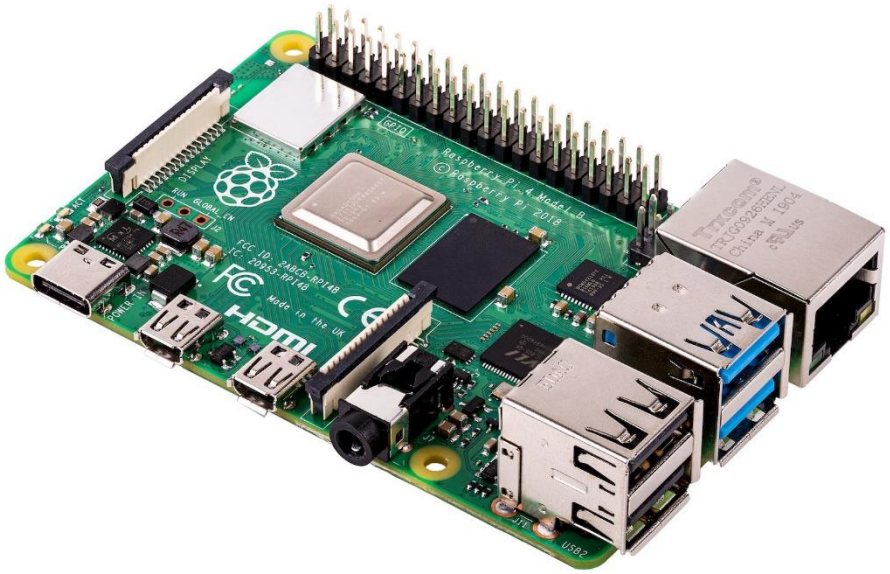


Рисунок 2.3 – Зовнішній вигляд мікрокомп’ютеру Raspberry Pi 4 B

Raspberry Pi 4 оснащений чотирьохядерним процесором Broadcom BCM2711B0 (Cortex A-72) з тактовою частотою 1,5 ГГц і має до 4 ГБ

оперативної пам'яті LPDDR4 SDRAM у топовій конфігурації - моделі, яку ми розглядали.

Raspberry Pi 4 може бути використаний для багатьох речей. Для початку, ентузіасти-аматори використовують плати Pi як медіацентри, файлові сервери, ретро-ігрові консолі, роутери та мережеві блокувальники реклами. Однак це лише невелика частина того, що можливо. Існують сотні проектів, де люди використовують Raspberry Pi для створення планшетів, ноутбуків, телефонів, роботів, розумних дзеркал, для фотографування на краю космосу, для проведення експериментів на міжнародній космічній станції.

Завдяки швидшому процесору Raspberry Pi 4, здатному декодувати відео у форматі 4K, швидшому збереженню даних через USB 3.0 та швидшому мережевому з'єднанню через справжній гігабітний Ethernet, двері відкриті для багатьох нових застосувань. Це також перший Raspberry Pi, який підтримує два екрани одночасно – до двох дисплеїв з роздільною здатністю 4K – знахідка для творчих людей, яким потрібно більше простору на робочому столі.

Raspberry Pi можна використовувати як бюджетний настільний комп'ютер, а з виходом Raspberry Pi 4 він став ще кращим. Найбільша перевага для повсякденного використання - офісні програми, перегляд веб-сторінок, доступ до онлайн-сервісів - це додаткова пам'ять.

З 4 ГБ оперативної пам'яті Raspberry Pi 4 більше не бореться з важкими веб-сторінками та додатками і може без затримок перемикатися між повноцінними онлайн-сервісами, такими як Google's G Suite, та сучасними сайтами, навантаженими JavaScript. У багатьох відношеннях він мало чим відрізняється від ПК, що коштує в рази дорожче – завдяки покращеним характеристикам і легкому, але потужному робочому столу Raspbian [15]. Характеристики Raspberry Pi 4 B наведені у табл. 2.3 [16].

Таблиця 2.3 – Характеристики Raspberry Pi 4 B

Процесор	Cortex-A72
Тактова частота	1.5GHz
Кількість ядер	4
RAM	8 Гб
Стандарти бездротового зв'язку	Wi-Fi 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet
Робоча температура	0ºC – 50ºC

2.1.3 Вибір плати для серверу нотифікації

Raspberry Pi Zero 2 W (рис. 2.4) – це мініатюрний одноплатний комп'ютер вартістю 15 доларів, який ідеально підходить для проектів Інтернету речей та робототехніки. Він оснащений RP3A0, який включає 64-бітний чотирьохядерний процесор з тактовою частотою 1 ГГц і 512 МБ оперативної пам'яті.

Рисунок 2.4 – Зовнішній вигляд мікрокомп’ютеру Raspberry Pi 4 B

Raspberry Pi Zero 2 W виділяється своїми милими об’ємами. З розміром приблизно 65 мм на 30 мм, він має той самий форм-фактор, що і попередня модель Raspberry Pi Zero. Крім того, ця модель також оснащена знайомим 40-контактним роз'ємом GPIO, присутнім на повнорозмірних Raspberry Pi SBC. Сумісність контактів GPIO, зокрема, дозволяє використовувати великий

Raspberry Pi для полегшення створення прототипів. У готовому проекті ви можете замінити великий SBC на менший Raspberry Pi Zero 2 W, щоб зменшити його загальний розмір і енергоспоживання.

Нова плата Raspberry Pi поставляється в комплекті з системою RP3A0, яка має чотирьохядерний 64-бітний процесор ARM Cortex-A53 з тактовою частотою 1 ГГц, а система містить 512 МБ оперативної пам'яті SDRAM. Ці характеристики роблять новий Raspberry Pi Zero 2 W в п'ять разів швидшим за свого попередника.

Крім того, Raspberry Pi Zero 2 W оснащений вбудованим модулем Wi-Fi, що робить його підключення до домашньої мережі простішим, ніж будь-коли раніше. Це означає, що вам не потрібно додавати жодних зовнішніх частин до ваших IoT-проектів, щоб підключити їх до мережі. Окрім вищезгаданого 40-контактного роз'єму GPIO, інші варіанти підключення включають роз'єм mini-HDMI, один роз'єм mini-USB і стандартний інтерфейс камери Raspberry Pi [17].

Характеристики Raspberry Pi Zero 2 W наведені у табл. 2.4 [18].

Таблиця 2.4 – Характеристики Raspberry Pi Zero 2 W

Процесор	Broadcom BCM2710A1
Тактова частота	1 ГГц
Кількість ядер	4
RAM	512 Мб
Стандарти бездротового зв'язку	Wi-Fi 802.11b/g/n wireless LAN, Bluetooth 4.2, BLE
Робоча температура	-20ºC – 70ºC

2.1.4 Вибір акустичного датчику

Ультразвуковий датчик відстані HC-SR04 (рис. 2.5) – це датчик, який використовується для визначення відстані до об'єкта за допомогою сонара. Він ідеально підходить для проектів, які вимагають уникання об'єктів, визначення відстанні до них або визначення перешкод.

Рисунок 2.5 – Зовнішній вигляд датчику HC-SR04

HC-SR04 використовує безконтактний ультразвуковий сонар для вимірювання відстані до об'єкта і складається з двох ультразвукових передавачів, приймача і схеми управління. Передавачі випромінюють високочастотний ультразвуковий звук, який відбивається від будь-яких твердих об'єктів поблизу, а приймач слухає зворотне відлуння. Це відлуння потім обробляється схемою управління для обчислення різниці в часі між сигналом, що передається і приймається. HC-SR04 є чудовим пристроєм, оскільки він має низьку вартість, може живитися через 5В вихід і найкраще працює в діапазоні від 2 см до 400 см в межах 30-градусного конуса і має точність до 0,3 см. Характеристики датчику HC-SR04 наведені у табл. 2.5 [19].

Таблиця 2.5 – Характеристики датчику HC-SR04

Напруга живлення	5 В
Струм	20 мА
Робоча температура	від -15°C до 70°C
Ефективний робочий кут	15°
Кут зчитування	30°
Відстань зчитування	від 2 см до 400 см

Ціна	50 грн

Ультразвуковий герметичний датчик відстані JSN-SR04T (рис. 2.6) для проєктів складається з рознесених плати-приймача та герметичного передавача на кабелі. Генерує звукові хвилі, які відбиваються від об'єкта і повертатися в приймач; і за витраченим часом визначає відстань до об'єкта.

Рисунок 2.6 – Зовнішній вигляд датчику JSN-SR04T

Функціонально JSN-SR04T є аналогом популярного ультразвукового датчика відстані HC-SR04, зокрема сумісний з ним за алгоритмом роботи та обміну даними, що дає змогу замінювати ці датчики один одним без перепрошивки мікроконтролера. Водонепроникний датчик JSN-SR04T можна використовувати в роботах, системах автоматики, що працюють у несприятливих умовах або на вулиці, в автомобільних парктроніках тощо. Характеристики датчику HC-SR04 наведені у табл. 2.6 [20].

Таблиця 2.6 – Характеристики датчику JSN-SR04T

Напруга живлення	5 В
Струм	30 мА
Струм очікування	5 мА
Ефективний робочий кут	<50°

Відстань зчитування	від 25 см до 500 см
Ціна	218 грн

Makeblock Me Ultrasonic Sensor V3 (рис. 2.7) – це електронний модуль для вимірювання відстані в діапазоні від 3 см до 400 см. Його можна використовувати для проектів пов'язаних з вимірюванням відстані та об'їздом перешкод. В модуль вбудовано бібліотеку Arduino, захист від стрибків напруги, підтримується підключення через інтерфейс 6 Pin RJ25, а також є світлодіодний індикатор статусу для отримання інформації під час роботи.

Рисунок 2.7 – Зовнішній вигляд датчику Makeblock Me Ultrasonic Sensor V3

Ультразвуковий датчик поставляється з двома "вічками", одне з яких є ультразвуковим передавачем, а інше – ультразвуковим приймачем. Ультразвуковий передавач запускає ультразвукову хвилю в певному напрямку і починає відлік часу з моменту запуску. Ультразвукова хвиля поширюється в повітрі, негайно повертається назад, коли зустрічає перешкоди на своєму шляху, і зупиняє відлік часу негайно, коли ультразвуковий приймач отримує відбиту хвилю.

Характеристики датчику Makeblock Me Ultrasonic Sensor V3 наведені у табл. 2.7 [21].

Таблиця 2.7 – Характеристики датчику Makeblock Me Ultrasonic Sensor V3

Напруга живлення	5 В
Струм	30 мА
Ефективний робочий кут	30°
Ультразвукова частота	42 кГц
Відстань зчитування	від 3 см до 400 см
Ціна	880 грн

2.2 Опис програмного середовища Fritzing

Fritzing (рис. 2.8) – це програмне середовище для проектування і макетування схем. Програма Fritzing призначена для розробки електронних пристроїв від прототипу в вигляді макетної плати до кінцевого продукту у вигляді друкованої плати.

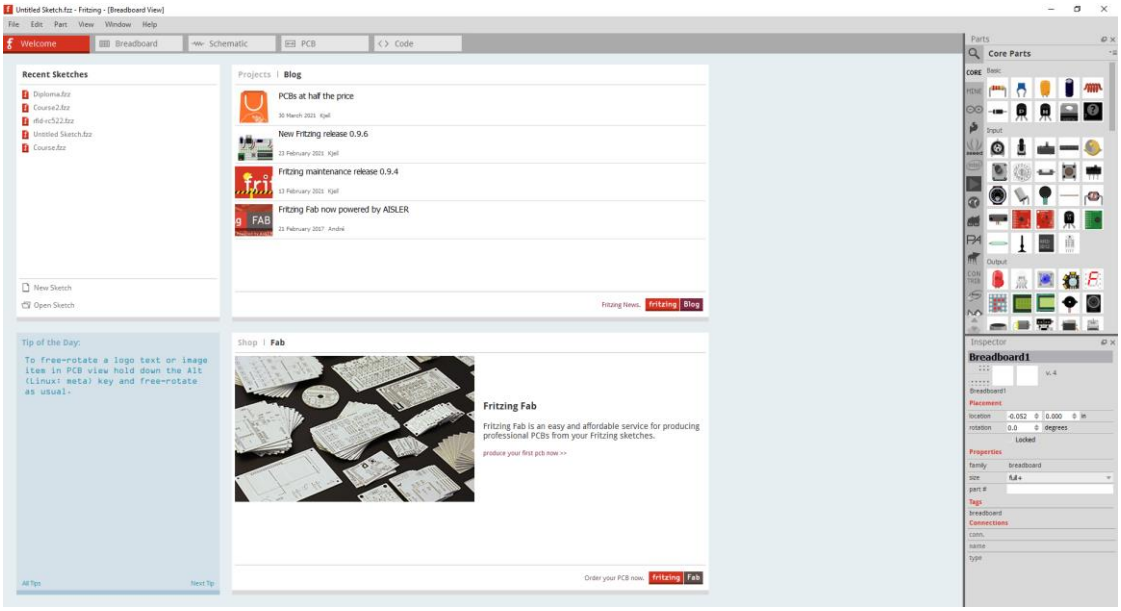


Рисунок 2.8 – Інтерфейс програмного середовища Fritzing

Опис основних елементів інтерфейсу програмного забезпечення Fritzing:

1. Вкладка «Welcome» – перша вкладка, яка відкривається при запуску програмного забезпечення, на якій зображені підказки щодо використання

програми, останні новини, а також вікно із нещодавно використаними схемами.

2. Вкладка «Breadboard» (рис. 2.9) – це вікно макетної плати, яке дозволяє створювати макетні схеми. Розмір макетної плати можна регулювати, а налаштування є інтерактивним, що дозволяє виводити компоненти і дроти на окремі виводи.

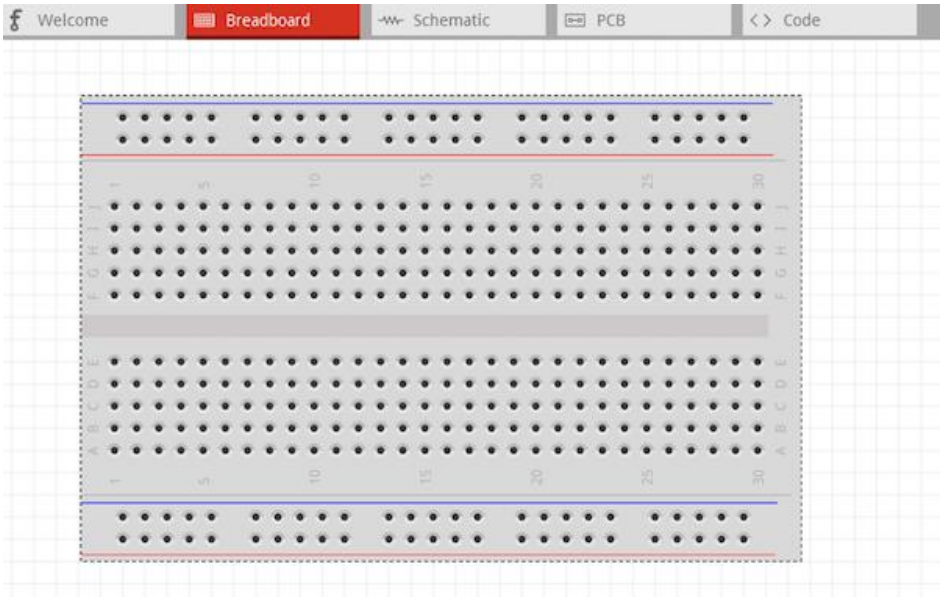


Рисунок 2.9 – Вкладка макетної плати

Вікно в правій частині екрану дозволяє знаходити і розміщувати на макетній платі різноманітні компоненти, а функція пошуку полегшує знаходження конкретних деталей (рис. 2.10). Для того, щоб помістити їх на схему достатньо вибрати зі списку і перетягнути на робочий простір лівою кнопкою мишки.

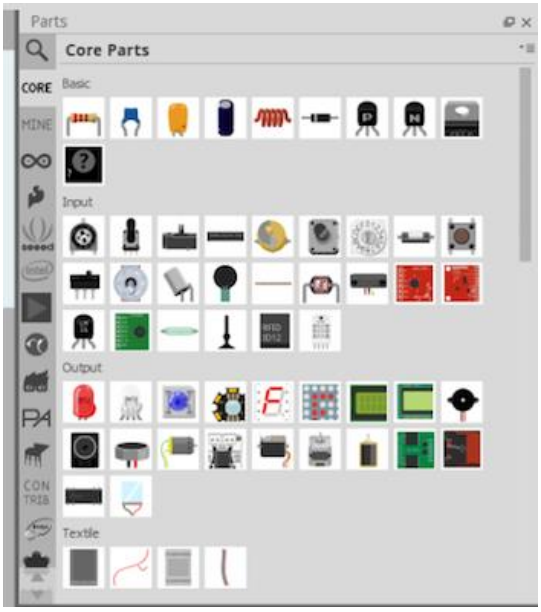


Рисунок 2.10 – Інтерфейс програмного середовища Fritzing

3. Вкладка «Schematic» (рис. 2.11) – вкладка, в якій створюється принципова схема проекту. Після того, як схема створена на макеті, її можна конвертувати в принципову схему. Однією з особливостей Fritzing є те, що коли користувач створює схему макетної плати, в схему автоматично додаються частини і формує між ними з'єднання відповідно до їх розміщення на макетній платі. Схема, проте, не буде достатньо акуратною або повною і вимагає деякої перестановки компонентів для формування з'єднань.

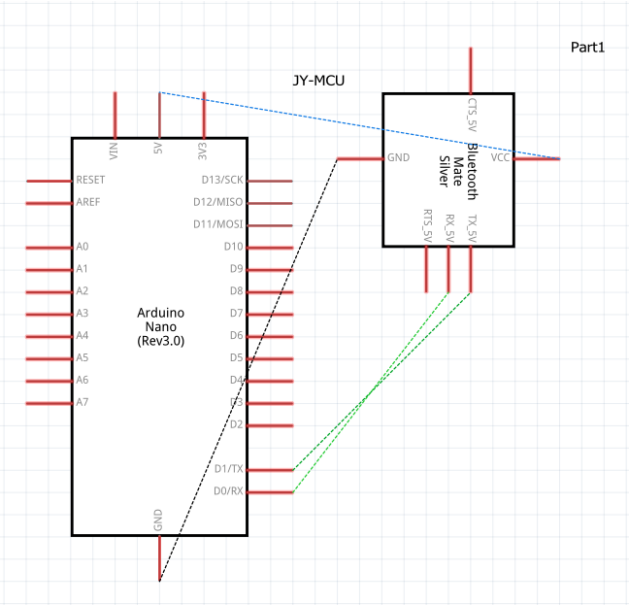


Рисунок 2.11 – Вкладка «Schematic»

4. Вкладка «PCB» (рис. 2.12) – вкладка, в якій відбувається створення друкованої плати. Компонування друкованої плати - це останній етап створення схеми у Fritzing. Як і на етапі створення схеми, деталі на друкованій платі автоматично розміщуються на платі. Знову ж таки, користувачеві потрібно правильно розташувати усі компоненти. Етап друкованої плати є необов'язковим, оскільки не в кожному проекті буде використовуватися друкована плата. Однією з головних переваг використання Fritzing є те, що програмне забезпечення працює в поєднанні з сервісом Fritzing PCB, який легко перетворює дизайн на Fritzing в недорогу друковану плату.

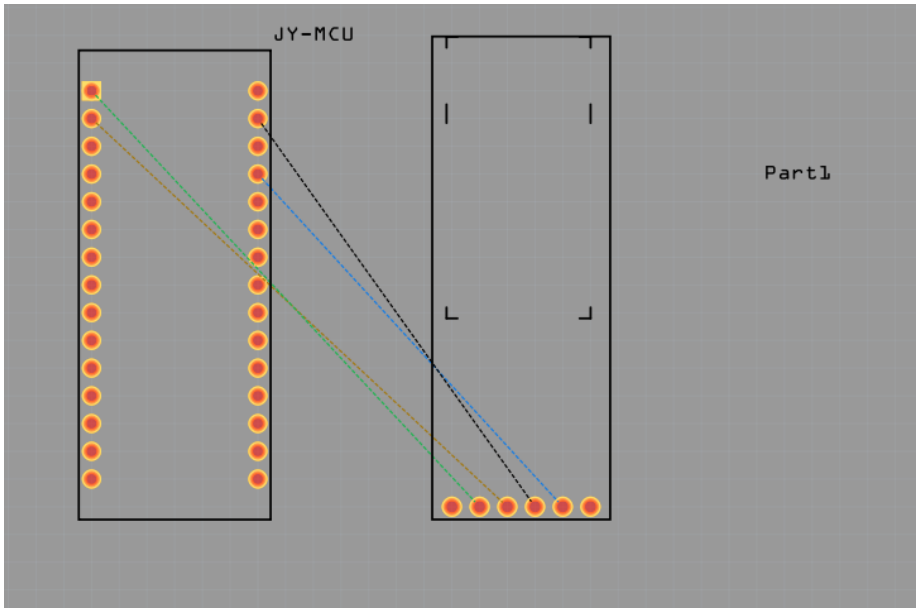


Рисунок 2.12 – Вкладка «PCB»

5. Вкладка «Code» (рис. 2.13) – вкладка, в якій відбувається написання скетчу. Зліва знизу вікна програми знаходяться дії, які відповідають за відкриття, створення та збереження файлу з вихідним кодом. Справа знизу розташовані меню вибору необхідної платформи, цільової плати, комунікаційного порту, серійного монітору, а також кнопка завантаження коду на приєднаний мікроконтролер.

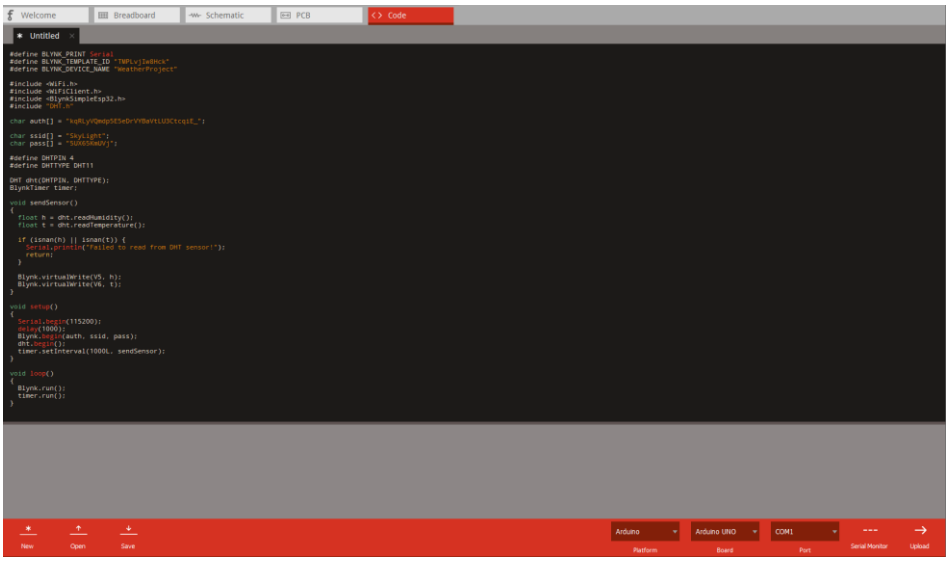


Рисунок 2.13 – Вкладка «Code»

6. Вікно «Parts» (рис. 2.14) – вікно, у котрому зображено список готових, розподілених за категоріями компонентів, що має вбудовану функцію пошуку. Також є можливість імпорту та експорту необхідних компонентів.

Рисунок 2.14 – Вікно «Parts»

7. Вікно «Inspector» (рис. 2.15) – вікно, в якому показано як саме виглядає компонент, його позначення на принципових схемах та властивості.

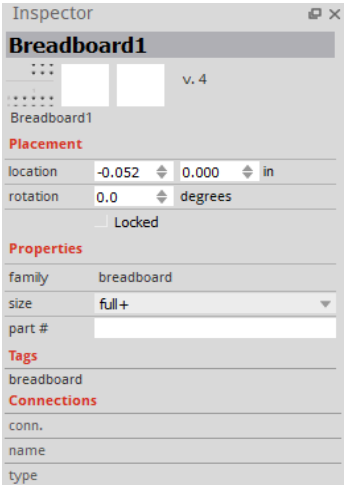


Рисунок 2.15 – Вікно «Inspector»

2.3 Огляд програмного середовища Arduino IDE

Програмне забезпечення Arduino IDE – найпопулярніша платформа аматорської та освітньої електроніки і робототехніки, що дозволяє легко писати код і завантажувати його на будь-яку плату.

Інтерфейс зображено на рис. 2.16, основою ПЗ є мова С ++. Структура Arduino IDE є простою, а отже може гарантувати швидке освоєння програми і перехід до розробки програм для плат Arduino.

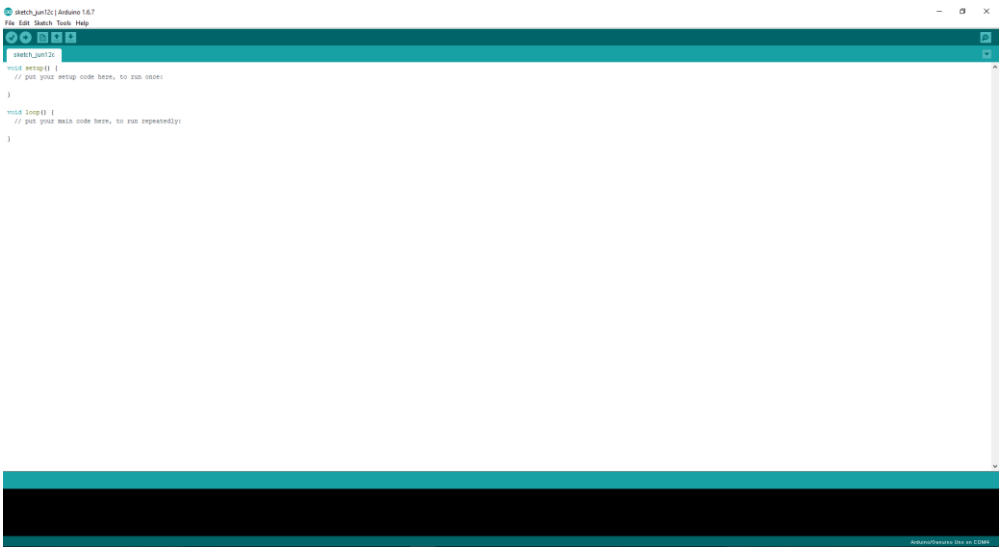


Рисунок 2.16 – Інтерфейс Arduino IDE

Переваги середовища розробки Arduino IDE:

– вивчення Arduino допоможе розібратися із С ++;

– можливість використання команд та функцій C++ аналогічних командам Arduino;

– можливість живлення і програмування плати Arduino за допомогою одного USB кабелю;

– можливість розробки простого за структурою проекту за кілька хвилин, використовуючи для цього лише стандартні бібліотеки. Для роботи з багатьма датчиками, сенсорами та іншими додатковими модулями існують написані бібліотеки, для використання яких непотрібно великого досвіду в програмуванні;

– досить гарне налаштування послідовних і SPI інтерфейсів зв'язку.

Недоліки середовища розробки Arduino IDE:

– щоб закінчити проект із застосуванням Arduino, доведеться вручну прошити завантажувач в кожен новий мікроконтролер ATmega. Він займає 2 Кб пам'яті;

– відсутність простого способу зміни тактової частоти, модель 3,3 / 8 МГц може спокійно працювати на частоті 12 МГц;

– при переповненні ISR таймера переривання відбувається кожні 16 K тактів в фоновому режимі, що зроблено для функцій millis () і micros (), але працює навіть коли вони не використовуються;

– порожній проект Arduino займає 466 байт на Arduino UNO і 666 байт на Arduino Mega2560.

Висновки до розділу 2

В цьому розділі було проведено огляд компонентів апаратно-програмного комплексу, а саме вибір мікроконтролера системи, вибір мікрокомп’ютеру для серверу бази даних, вибір плати для серверу нотифікації, вибір акустичного датчику. Також розглянуто основні програмні забезпечення, що були використанні при створенні системи, їх особливості, а також переваги та недоліки.

3 РОЗРОБКА АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Алгоритм роботи системи визначення джерела звуку

Система визначення місця знаходження джерела звуку складається із mesh-мережі акустичних комплексів, сегменту сховища, а також сегменту нотифікації. До сегмента сховища входить пул серверів, що відповідають за зберігання даних, створених після активації акустичного комплексу. Сегмент нотифікації складається із серверів, які відповідають за відправку сповіщення усім клієнтам на мобільний додаток після активації акустичного комплексу mesh-мережі системи. Для розподілення мережевого трафіку між кількома серверами, збільшення пропускної здатності та підвищення відмовостійкості системи використовується пристрій, який працює як зворотний проксі-сервер – балансувальник навантаження.

3.2 Проектування концептуальної схеми апаратно-програмного комплексу

Концептуальна схема використовується для візуального представлення того, як саме працює система за допомогою графічного відображення розташування та взаємозв'язків ключових атрибутів у системі. Перш за все концептуальна схема системи визначення положення джерела звуку складається із об’єднаних у mesh-мережу акустичних комплексів (рис. 3.1). Акустичні комплекси в свою чергу містять мікроконтролер ESP32, камеру та активний звуковий модуль, який виступає в якості сигналізації.

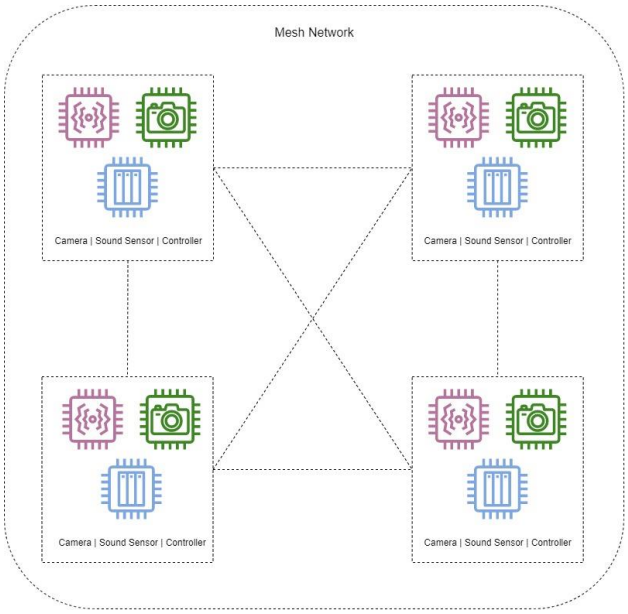


Рисунок 3.1 – Mesh-мережа акустичних комплексів

Наступний етап – сегмент сховища, що зображено на рис. 3.2. В цьому сегменті балансувальник навантаження розподіляє мережевий трафік між N серверами, кожен з яких звертається до бази даних, де зберігаються дані, що створюються під час активації акустичного комплексу – часова мітка, інформація, що стосується створених знімків, а також дані щодо активованого акустичного комплекса.

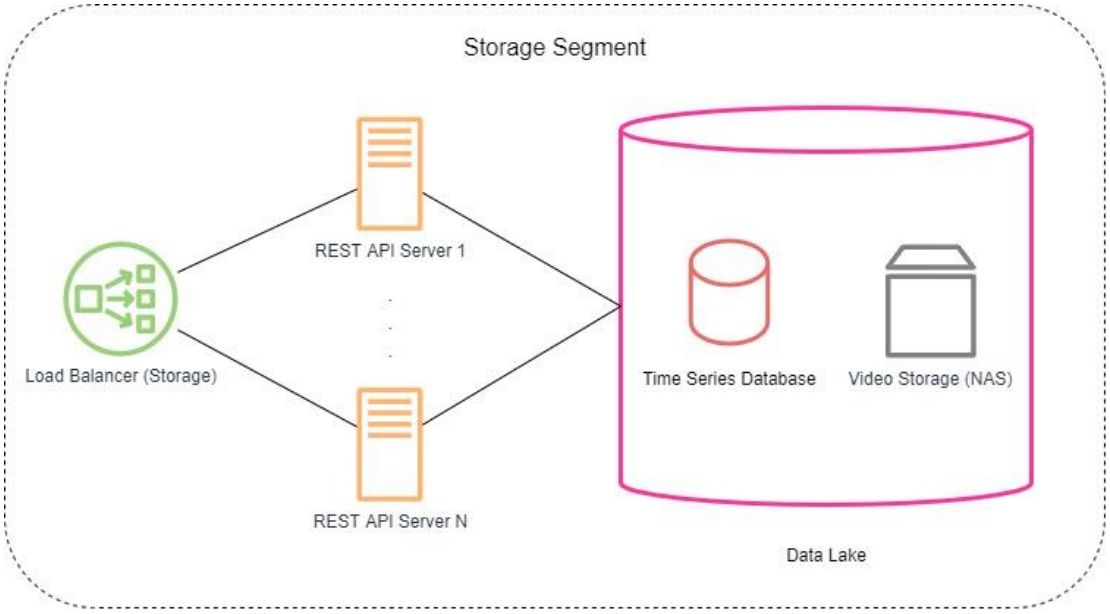


Рисунок 3.2 – Сегмент сховища

На рис. 3.3 зображено сегмент нотифікації. В цьому сегменті балансувальник навантаження розподіляє мережевий трафік між N серверами, які відповідають за відправку сповіщення усім клієнтам на мобільний додаток після активації акустичного комплексу mesh-мережі системи.

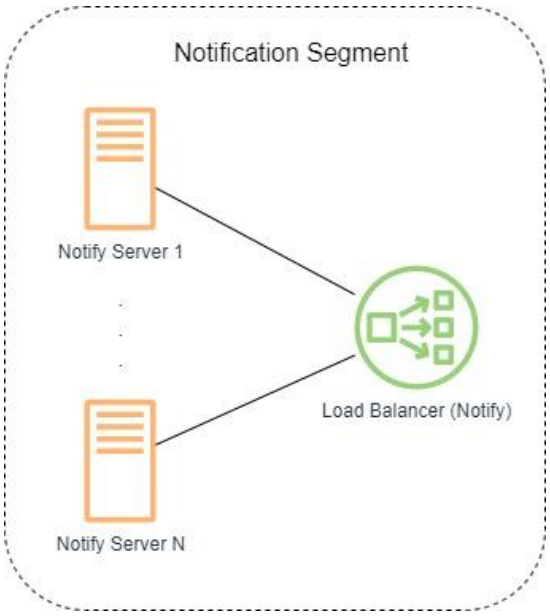


Рисунок 3.3 – Сегмент нотифікації

Системи сповіщення (рис. 3.4) необхідні для інформування про порушення норми звуку на девайси користувачів.

Рисунок 3.4 – Системи сповіщення

Відділ аналітики (рис. 3.5) – дані які використовуються для аналізу, аналітики та подальшої обробки отриманих даних для покращення безпеки у приміщеннях, оптимізації діяльності та підвищення ефективності.

Рисунок 3.5 – Відділ аналітики

Інші системи – визначають можливості для розширення системи, наприклад підключення додаткових засобів вимірювання безпеки, збору інформації, активації, системи аутентифікаціїї, збору даних та обчислювальні системи (рис. 3.6).

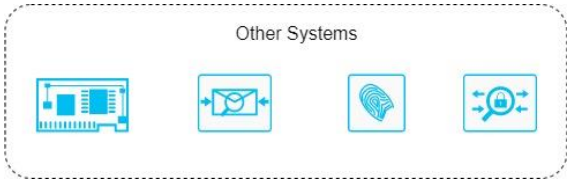


Рисунок 3.6 – Інші системи

Ознайомитися із повною концептуальною схемою апаратно-програмного комплексу можна у додатку А.

3.3 Створення блок-схеми алгоритму акустичного комплексу

На рис. 3.7 зображено блок-схему алгоритму акустичного комплексу. Спочатку здійснюється зчитування звукових хвиль у просторі, після чого контролер перевіряє чи відбулось порушення норми звуку. В тому випадку, якщо звук порушив визначені межі норми, то далі відбувається створення знімку камерою. На наступному етапі дані відправляються на Rest Api сервер та сервер нотифікації. Коли Rest Api сервер отримає інформацію з акустичного кластеру, відбувається зберігання знімків у сховище NAS, після чого у базу даних зберігаються дані з акустичного датчику та метадані

знімків. Коли сервер нотифікації отримає інформацію з акустичного кластеру, відбувається відправка сповіщення на девайси користувачів.

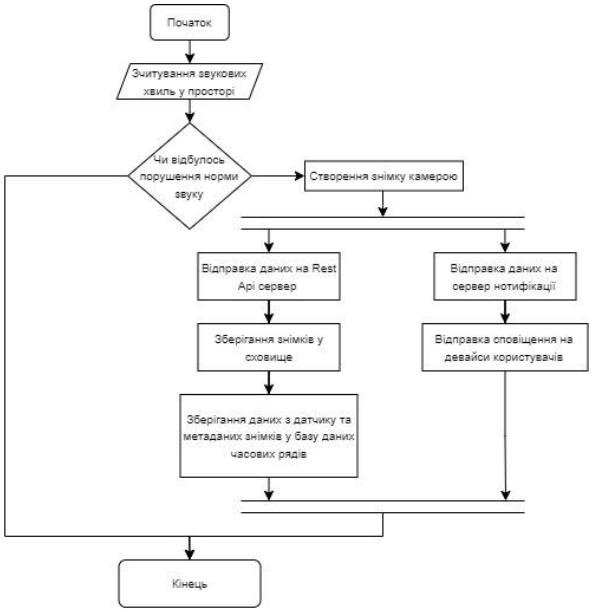


Рисунок 3.7 – Блок-схеми алгоритму акустичного комплексу

3.4 Проектування пристрою на основі ESP32 та акустичних датчиків HC-SR04

Акустичний комплекс складається із мікроконтролеру ESP32-CAM, датчику HC-SR04, а також активного звукового модулю. Підключаємо ультразвуковий датчик до ESP32-CAM – підключаємо триггер пін до піну 13, а ехо до піну 15, а також активний звуковий модуль до піну 14. На рис. 3.8 зображено макетну схему і на рис. 3.9 принципову схему акустичного комплексу.

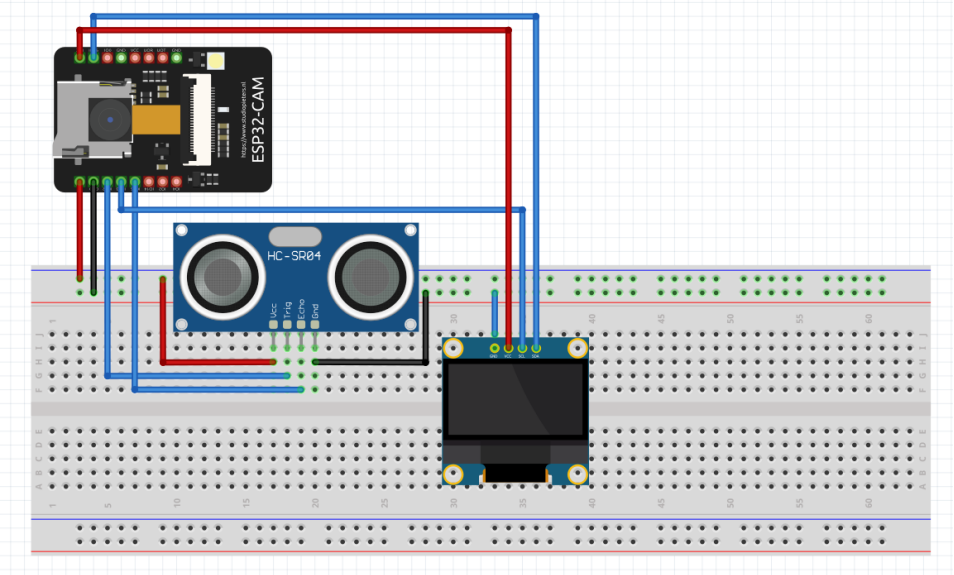


Рисунок 3.8 – Макетна схема акустичного комплексу

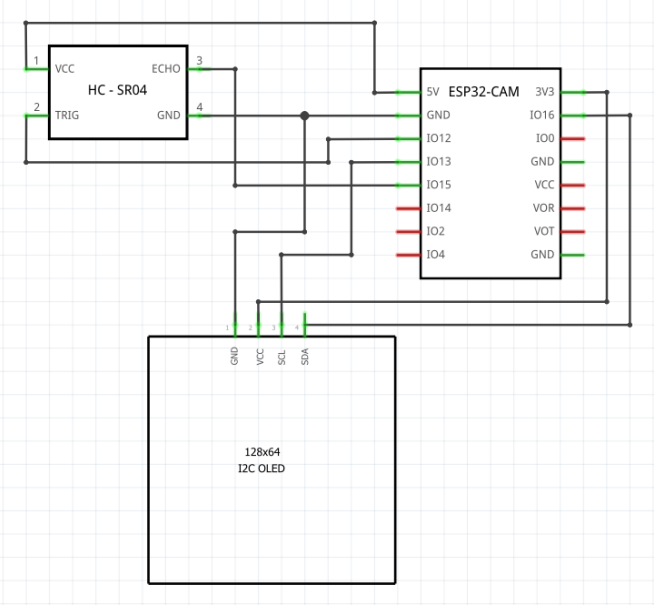


Рисунок 3.9 – Принципова схема акустичного комплексу

3.5 Розробка діаграми послідовності системи визначення місця знаходження джерела звуку

Діаграма послідовності використовується для моделювання логіки сервісів, проектування поведінки системи та відображення того, як певні завдання виконуються між користувачами та системою, зображуючи процеси, що відбуваються у системі, а також послідовність повідомлень, якими вони обмінюються, що організовується за об'єктами і часом.

На рис. 3.10 зображено діаграму послідовності, на якій було розміщено такі об’єкти – акустичний датчик, камеру, контролер, Rest Api сервер, базу даних, відеосховище, систему репортингу, сервер нотифікації та смартфон.

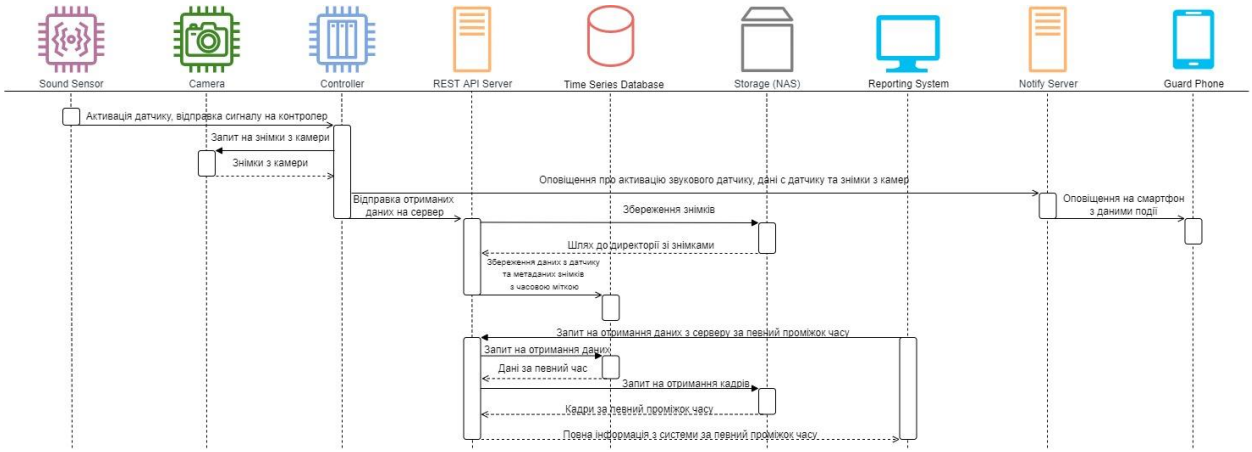


Рисунок 3.10 – Діаграма послідовності апаратно-програмного комплексу

Після активації акустичного комплексу відбувається відправка сигналу на контролер, який, в свою чергу, робить запит на знімки з камери. Після отримання знімків, контролер відправляє отриманні дані на Rest Api сервер та сервер нотифікації. Rest Api сервер зберігає дані з датчику та метадані знімків з часовою міткою у базу даних, а самі знімки у відеосховище. Система репортингу робить запит на отримання даних з серверу за певний проміжок часу, після чого сервер робить запит до бази даних та відеосховища, і, отримавши дані відправляє повну інформацію до системи репортингу. Сервер нотифікації, після отримання даних від контролеру відправляє сповіщення з даними про подію на девайси користувачів.

3.6 Розробка структурної діаграми системи визначення місця знаходження джерела звуку

Структурна діаграма використовується для графічного відображення логічних компонентів системи у розподіленій мережі. На рис. 3.11 зображено частину структурної діаграми усієї системи, а саме структурну діаграму акустичної mesh-мережі, що складається із декількох акустичних комплексів, в кожному з яких мікроконтролер ESP32-Cam з’єднується із акустичним

датчиком HC-SR04 та пє'зо-модулем за допомогою інтерфейсу GPIO – інтерфейсу введення/виведення загального призначення.

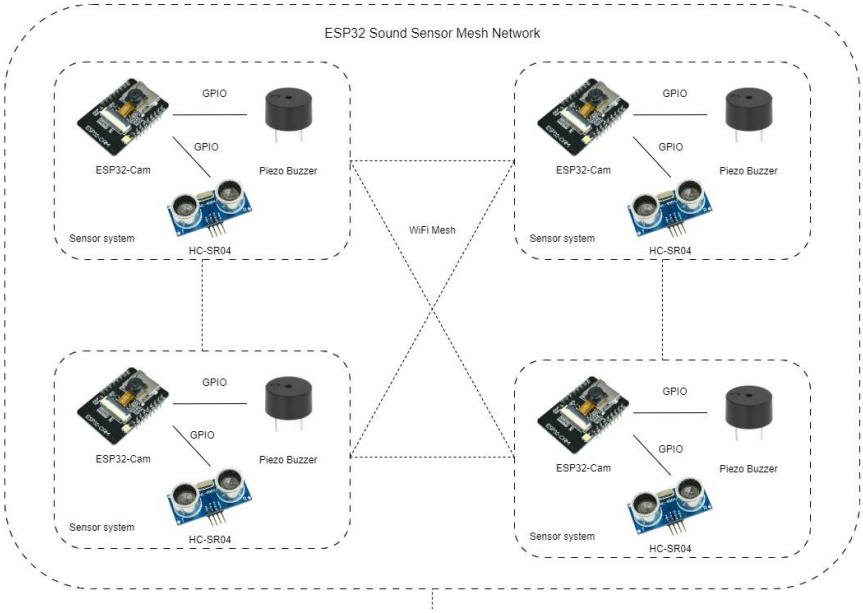


Рисунок 3.11 – Структурна діаграма акустичної mesh-мережі

Далі, на рис. 3.12 зображено ядро системи – маршрутизатор Xiaomi Mesh System. У mesh-мережі, створеній за допомогою цього маршрутизатора у разі виявлення несправного вузла вся мережева архітектура перепланується для забезпечення нормальної роботи Mesh-мережі. Технологія OFDMA дає змогу роутерам обробляти потоки даних із декількох пристроїв одночасно, коли великій кількості пристроїв потрібне передавання даних, ефективно знижуючи навантаження на мережу та значно скорочуючи затримку [22]. Також зображено балансувальники навантаження та сегмент користувачів.

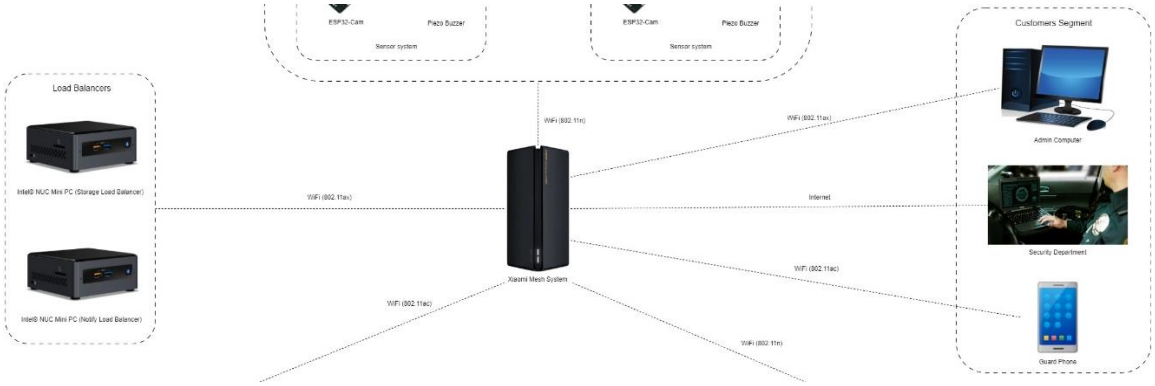


Рисунок 3.12 – Структурна діаграма системи

Структурна діаграма сегменту сховища (рис. 3.13) складається із декількох плат Raspberry Pi 4 Model B, що використовуються для Rest Api серверу фронтенд серверу та балансувальника навантаження, а також сховища NAS QNAP 2BAY TS-233, що під’єднуються до мережевого комутатора TP-Link TL-SG105 для з'єднання пристроїв в комп'ютерній мережі.

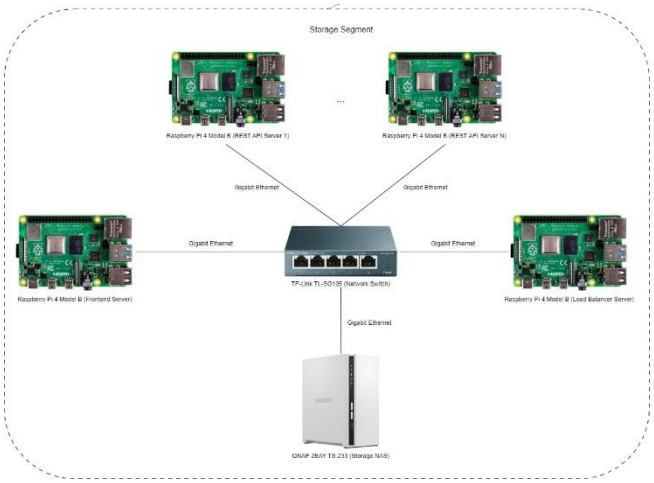


Рисунок 3.13 – Структурна діаграма сегменту сховища

Структурна діаграма сегменту нотифікації (рис. 3.14) складається із декількох плат Raspberry Pi Zero 2 W, що використовуються для серверу сповіщення.

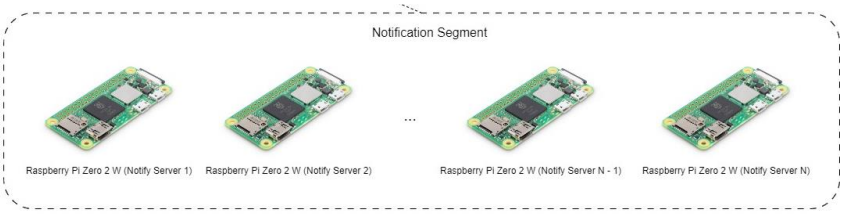


Рисунок 3.14 – Структурна діаграма сегменту нотифікації

Повна структурна діаграма апаратно-програмного комплексу знаходиться у додатку Б.

3.7 Розробка діаграми програмної архітектури системи визначення місця знаходження джерела звуку

Діаграми програмної архітектури використовуються для візуального представлення програмних компонентів і систем, які допомагають краще зрозуміти архітектуру та дизайн проектованої системи.

На рис. 3.15 зображено програмну частину акустичного комплексу. Програмний код системи написано на мові програмування, що використовується в Arduino IDE, що базується на широко використовуваній і відомій мові програмування C++. Програмну частина акустичного комплексу взаємодіє із програмною частиною балансувальника навантаження за допомогою протоколу HTTPS, що використовує шифрування для безпечного зв'язку через комп'ютерну мережу і широко використовується в Інтернеті [23].

Рисунок 3.15 – Програмна частина акустичного комплексу

Програмну частину балансувальників навантаження для серверу нотифікації та Rest Api серверу (рис. 3.16), а також для бази даних, сховища та фронтенду (рис. 3.17) реалізовано за допомогою програмного забезпечення HAProxy. HAProxy - це безкоштовний, швидкий і надійний проксі, що забезпечує високу доступність, балансування навантаження і проксі для додатків на основі TCP і HTTP [24]. Програмна частина балансувальника навантаження взаємодіє із програмною частиною акустичного комплексу за допомогою протоколу HTTPS.

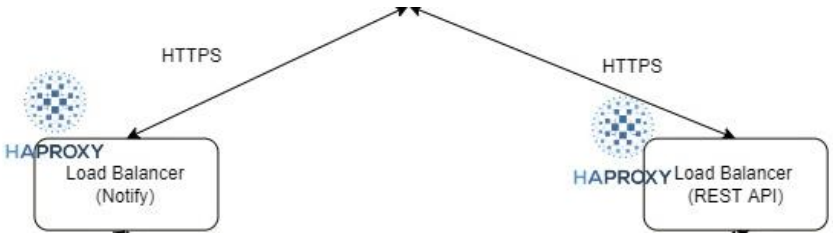


Рисунок 3.16 – Програмна частина балансувальника навантаження для серверу нотифікації та Rest Api серверу

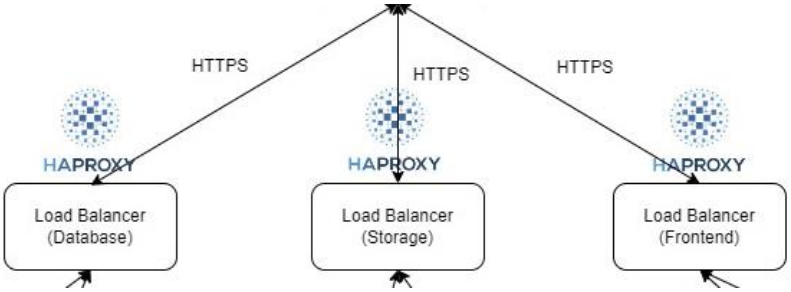


Рисунок 3.17 – Програмна частина балансувальника навантаження для бази даних, сховища та фронтенду

Програмну частину серверу нотифікації (рис. 3.18) реалізовано за допомогою мови програмування Elixir.

Elixir - це динамічна, функціональна мова для створення масштабованих та підтримуваних додатків, що працює на віртуальній машині Erlang, яка відома створенням розподілених та відмовостійких систем з низькою затримкою [25]. Програмна частина серверу нотифікації взаємодіє із програмною частиною балансувальника навантаження за допомогою протоколу HTTPS.

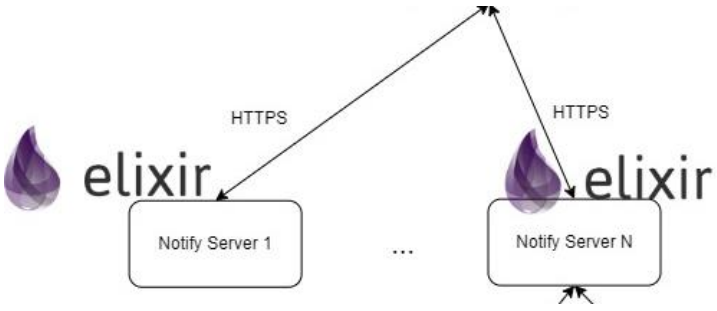


Рисунок 3.18 – Програмна частина серверу нотифікації

Програмну частину мобільного додатку (рис. 3.19) реалізовано за допомогою фреймворку Flutter. Flutter – це фреймворк з відкритим вихідним кодом від Google для створення красивих, нативно скомпільованих, багатоплатформних додатків за допомогою єдиної кодової бази. Код Flutter компілюється в машинний код ARM або Intel для швидкої роботи на будь-якому пристрої [26]. Програмна частина мобільного додатку взаємодіє із програмною частиною серверу нотифікації за допомогою протоколу WebSocket Secure.

WebSocket – це передова технологія, яка дозволяє відкривати двонаправлені інтерактивні сеанси зв'язку між браузером користувача і сервером. Його можна використовувати для надсилання повідомлень на сервер та отримання відповідей на основі подій замість того, щоб робити запити до сервісу. Так як WebSocket являється протоколом із підтримкою стану, це означає, що з'єднання між клієнтом і сервером буде залишатися відкритим, поки будь-яка зі сторін не розірве його. Для захищеного з’єднання необхідно використовувати захищенний протоколу wss://, замість незахищеного ws://. Як і HTTPS, WSS шифрується, що захищає від атак типу "man-in-the-middle". Різноманітні атаки на WebSockets стають неможливими, якщо протокол захищений [27].

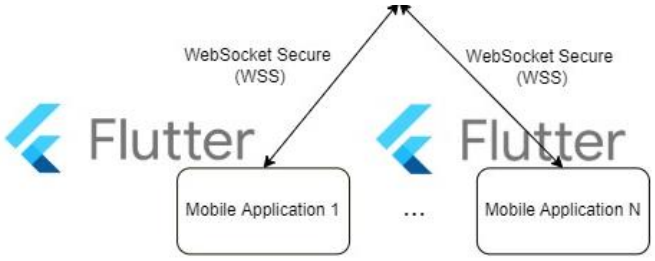


Рисунок 3.19 – Програмна частина мобільного додатку

Програмну частину Rest Api серверу (рис. 3.20) реалізовано за допомогою мови програмування Rust. Rust – це мова, що дає можливість створювати надійне та ефективне програмне забезпечення, завдяки надзвичайно швидкому та ефективному використанні пам'яті, можливості

працювати з критично важливими до продуктивності сервісами, запускатися на вбудованих пристроях та легко інтегруватися з іншими мовами. Багата система типів та модель власності Rust гарантують безпеку пам'яті та потоків, що дозволяє вам усунути багато типів помилок під час компіляції [28]. Програмна частина Rest Api серверу взаємодіє із програмною частиною балансувальника навантаження за допомогою протоколу HTTPS.

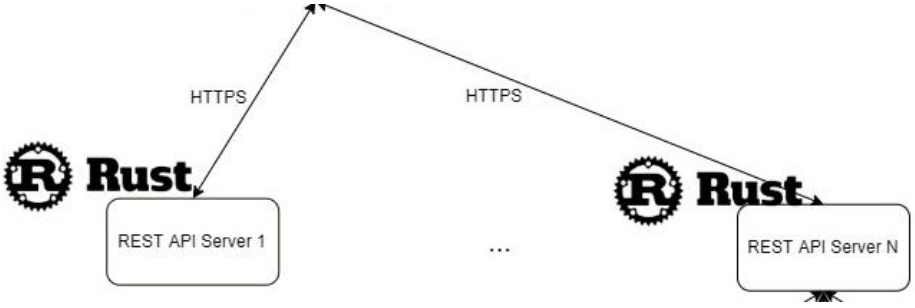


Рисунок 3.20 – Програмна частина Rest Api серверу

Програмну частину бази даних (рис. 3.21) реалізовано за допомогою InfluxDB. Інсталяція InfluxDB Enterprise дозволяє створити кластерну інсталяцію InfluxDB, яка складається з двох окремих програмних процесів: вузлів даних і мета-вузлів. Для запуску кластера InfluxDB потрібні як мета-вузли, так і вузли даних. Мета-вузли та вузли даних спілкуються один з одним за допомогою протоколу TCP Protobuf і групи консенсусу Raft. У кластері всі мета-вузли повинні взаємодіяти з усіма іншими мета-вузлами. Мета-вузли зберігають узгоджене уявлення про метадані, які описують кластер та можуть працювати на дуже скромних за розміром віртуальних машинах. Вузли даних реплікують дані і запитують один одного за допомогою протоколу Protobuf через TCP та відповідають за обробку всіх записів і запитів [29]. Програмна частина бази даних взаємодіє із програмною частиною балансувальника навантаження за допомогою протоколу HTTPS.

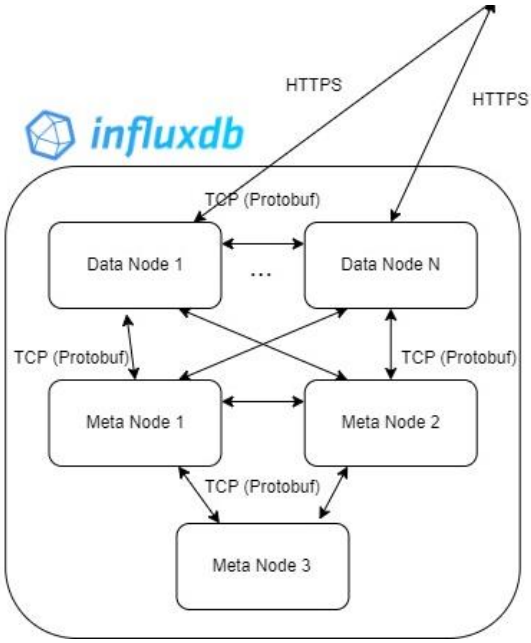


Рисунок 3.21 – Програмна частина бази даних

Програмну частину сховища (рис. 3.22) реалізовано за допомогою розгорнутого кластеру інстансів MinIO, які, в свою чергу, розгорнуті поверх кластеру K3s. MinIO пропонує високопродуктивне, сумісне зі стандартом S3 сховище об'єктів. Простота є основою для інфраструктури ексафлопексних даних - як з технічної, так і з операційної точки зору. MinIO - найшвидше у світі сховище об'єктів з опублікованими результатами GET/PUT, що перевищують 325 Гігабайт/с і 165 Гігабайт/с на 32 вузлах NVMe накопичувачів і мережі 100 Гбіт/с. K3s – це полегшений дистрибутив Kubernetes, який спрощує розгортання Kubernetes і містить лише основні компоненти Kubernetes, такі як kube-apiserver, kube-scheduler, kubelet, kube-controller-manager та kube-proxy. Використання лише основних компонентів робить дистрибутив легким, але ви можете легко замінити компоненти зовнішніми доповненнями. Він об'єднує ці компоненти в уніфіковані процеси, представлені у вигляді простої моделі сервера та агента. Коли запускається сервер k3s, він запускає сервер Kubernetes і автоматично реєструє localhost як агент для створення одновузлового кластера Kubernetes.

Програмна частина сховища взаємодіє із програмною частиною балансувальника навантаження за допомогою протоколу HTTPS.

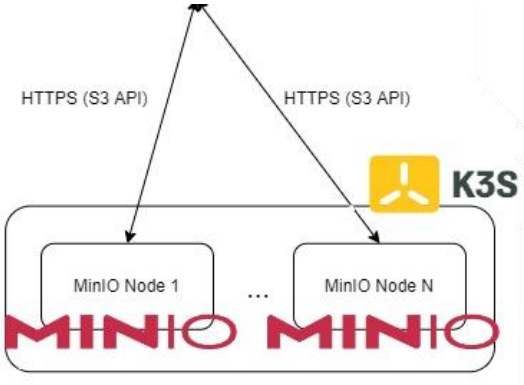


Рисунок 3.22 – Програмна частина сховища

Програмну частину фронтенду (рис. 3.23) реалізовано за допомогою фреймфорку Next.JS. Next.JS – це гнучкий фреймворк React, який надає вам блоки для створення швидких веб-додатків, що обробляє інструменти та конфігурацію, необхідні для React, а також надає додаткову структуру, функції та оптимізацію для вашого додатку. Можна використовувати React для створення інтерфейсу користувача, а потім поступово впроваджувати функції Next.js для вирішення загальних вимог додатків, таких як маршрутизація, отримання даних, інтеграція. Програмна частина фронтенду взаємодіє із програмною частиною балансувальника навантаження за допомогою протоколу HTTPS.

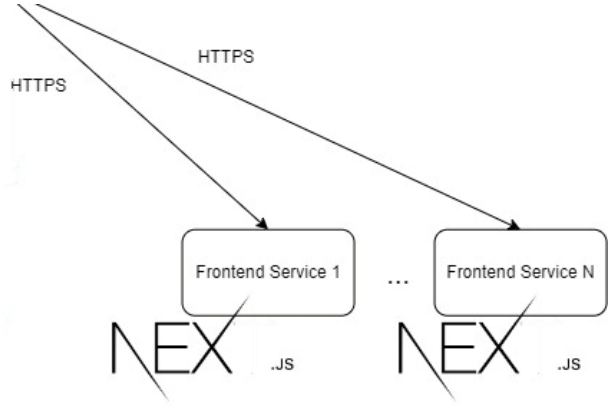


Рисунок 3.23 – Програмна частина фронтенду

Повна діаграма програмної архітектури апаратно-програмного комплексу знаходиться у додатку В.

3.8 Розробка програмного забезпечення

Спочатку потрібно підключити необхідні бібліотеки для роботи з камерою, створення веб-сервера та використання SPIFFS (рис. 3.24).

```
#include "esp_camera.h"
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "driver/rtc_io.h"
#include <ESPAsyncWebServer.h>
#include <StringArray.h>
#include <SPIFFS.h>
#include <FS.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
```

Рисунок 3.24 – Підключення необхідних бібліотек

Далі визначаємо виводи камери для модуля ESP32-CAM AI THINKER (рис. 3.25).

```
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
```

Рисунок 3.25 – Визначення виводів камери

На рис. 3.26 зображено визначення контактів для ультразвуку та зумера.

```
const int trigPin = 13;
const int echoPin = 15;
const int buzzPin = 14;
```

Рисунок 3.26 – Визначення контактів для ультразвуку та зумера

Наступним кроком підключаємо ESP32-CAM до локальної мережі (рис. 3.27).

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi...");
}
```

Рисунок 3.27 – Підключення ESP32-CAM до локальної мережі

Далі виконуємо ініціалізацію SPIFFS (рис. 3.28).

```
if (!SPIFFS.begin(true)) {
  Serial.println("An Error has occurred while mounting SPIFFS");
  ESP.restart();
}
else {
  delay(500);
  Serial.println("SPIFFS mounted successfully");
}
```

Рисунок 3.28 – Ініціалізація SPIFFS

У фунції loop, починаємо посилати ультразвуковий імпульс коли вивід тригера встановлюється у положення high на 10 мкс, вимірюємо відгук від ехо-контакту, а також знаходимо відстань до об’єкту (рис. 3.29).

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);
distance = duration * SOUND_SPEED/2;
```

Рисунок 3.29 – Знаходження відстані до об’єкту

При виявлені джерела звуку відбувається перевірка інтернет з’єднання, створення знімку, а також відправлення даних до сервера (рис. 3.30).

```
if (distance > 0) {
  if(WiFi.status()== WL_CONNECTED){
    date = timeClient.getFormattedTime()
    capturePhotoSaveSpiffs();
    printInfo(distance, date);

    WiFiClient client;
    HTTPClient http;

    http.begin(client, serverName);
    File photo = SPIFFS.open(FILE_PHOTO, "r");
    if (!photo) {
      Serial.println("Failed to open file for reading");
      return;
    }
    client.write(photo);

    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
    String httpRequestData = "distance="+distance+"&sensor_number="+sensor_number+"&datetime="+date;
    int httpResponseCode = http.POST(httpRequestData);
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    http.end();
  }
  else {
    Serial.println("WiFi Disconnected");
  }
}
delay(1000);
```

Рисунок 3.30 – Відправлення даних до сервера

Функція capturePhotoSaveSpiffs() робить знімок фото та зберігає його у файловій системі флеш-пам'яті SPIFFS (рис. 3.31).

```
void capturePhotoSaveSpiffs( void ) {
  camera_fb_t * fb = NULL; // pointer
  bool ok = 0;

  do {
    Serial.println("Taking a photo...");

    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      return;
    }

    Serial.printf("Picture file name: %s\n", FILE_PHOTO);
    File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);

    if (!file) {
      Serial.println("Failed to open file in writing mode");
    }
    else {
      file.write(fb->buf, fb->len);
      Serial.print("The picture has been saved in ");
      Serial.print(FILE_PHOTO);
      Serial.print(" - Size: ");
      Serial.print(file.size());
      Serial.println(" bytes");
    }
    file.close();
    esp_camera_fb_return(fb);

    ok = checkPhoto(SPIFFS);
  } while ( !ok );
}
```

Рисунок 3.31 – Функція capturePhotoSaveSpiffs()

Функція checkPhoto() перевіряє, чи було успішно збережено фотографію у файловій системі флеш-пам'яті SPIFFS (рис. 3.32).

```
bool checkPhoto( fs::FS &fs ) {
  File f_pic = fs.open( FILE_PHOTO );
  unsigned int pic_sz = f_pic.size();
  return ( pic_sz > 100 );
}
```

Рисунок 3.32 – Функція checkPhoto()

3.9 Розробка мобільного додатку користувача

Перш за все перші скетчі та варіанти інтерфейсу були спроектовані за допомогою програмного забезпечення Figma. Приклад створеного інтерфейсу за допомогою програмного забезпечення Figma зображено на рис. 3.33.

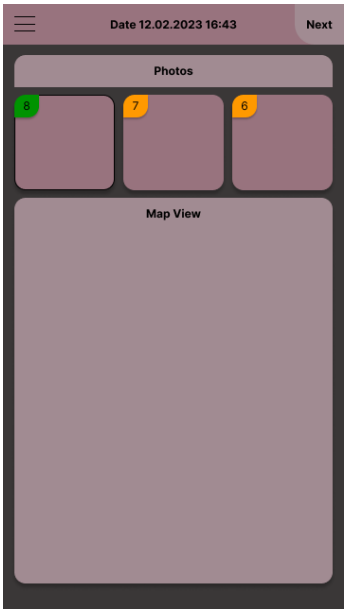


Рисунок 3.33 – Спроектований інтерфейс додатку у ПЗ Figma

На наступному етапі було розроблено мобільний додаток за допомогою фреймворку Flutter. На рис. 3.34 зображено інтерфейс створеного додатку за допомогою фреймворку Flutter.

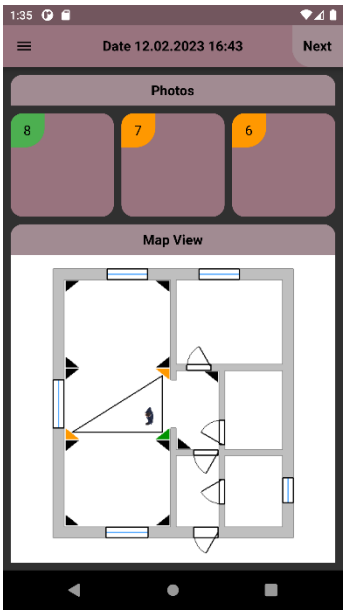


Рисунок 3.34 – Створений додаток за допомогою Flutter

Основна функція додатку – відображення звіту результату роботи mesh-мережі акустичних комплексів. На користувацькому інтерфейсі зображено мапу активації датчиків, на якій можна побачити місцезнаходження датчиків, якими було знайдено джерело звуку – зеленим кольором позначено найближчий до джерела датчик, помаранчевим – два інші датчики, за допомогою яких методом тріангуляції було виявлено місцезнаходження джерела звуку на мапі. Окрім цього розроблений додаток дозволяє побачити фотознімки з активованих датчиків із зазначенням їх персонального номеру та дату оформленого звіту.

Висновки до розділу 3

В цьому розділі було розглянуто алгоритм роботи системи визначення джерела звуку, створено блок-схему алгоритму та макетну і принципову схеми акустичного комплексу, концептуальну схему, розроблено діаграму послідовності, структурну діаграму, а також діаграму програмної архітектури системи визначення місця знаходження джерела звуку. Окрім цього розроблено програмне забезпечення апаратно-програмного комплексу та створено мобільний додаток користувача.

ВИСНОВКИ

Для того щоб почати роботу з системою виявлення джерел звуку за допомогою mesh-мережі на базі ESP32 та акустичних датчиків було визначено актуальність системи виявлення звуку, поставлено завдання для досягнення поставленої мети, було виконано порівняльний аналіз відомих технічних та програмних рішень, проведено огляд теоретичних відомостей стосовно принципів виявлення джерела звуку, поглиблено знання стосовно сфери SSL, принципи роботи систем SSL, а також основні методи SSL. Після цього, задля уникнення проблем при проектуванні систем локалізації з використанням бездротових акустичних мереж, було визначено практичні вимоги до систем виявлення джерела звуку, а саме економічна ефективність, особливості впровадження, відмовостійкість системи, масштабованість системи та похибки вимірювання.

На наступному етапі, перш ніж переходити до проектування апаратно-програмного комплексу було проведено аналіз доступних мікроконтролерів, датчиків, сенсорів та на основі зібраних даних обрано головний мікроконтролер системи, мікрокомп’ютер для серверу бази даних, плату для серверу нотифікації та акустичного датчику. Окрім цього було описано програмні забезпечення, які були використанні при проектуванні системи, розглянуто їх особливості, а також переваги та недоліки.

В процесі розробки системи визначення місця знаходження джерела звуку було створено алгоритм роботи системи, спроектовано схеми, що потрібні для покращення розуміння структури системи, а саме – блок-схему алгоритму, макетну і принципову схеми акустичного комплексу; концептуальну схему, діаграму послідовності, структурну діаграму, а також діаграму програмної архітектури апаратно-програмного комплексу. Після цього розроблено програмне забезпечення системи визначення джерела звуку та створено мобільний додаток користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Kaushik Sunder, Kapil Jain, Yuxiang Wang; Pat. US10880669B2 United States, Binaural sound source localization, assigned 19.

2. Wai C. Chu, Edward Dietz Crump; Pat. US10966022B1 United States, Sound source localization using multiple microphone arrays, assigned 18.

3. Maher R. C. Fundamentals of Audio Signals and Systems. 2018. С. 3–28.

4. Kraljevic L., Russo M., Stella M., Sikora M. Free-Field TDOA-AOA Sound Source Localization Using Three Soundfield Microphones. *IEEE Access*. 2020. Вип. 8. С. 87749–87761.

5. Chen X., Wang G., Ho K. C. Semidefinite relaxation method for unified near-Field and far-Field localization by AOA. *Signal Processing*. 2021. Вип. 181. С. 107916.

6. Bouras C., Gkamas A., Kokkinos V., Papachristos N. Time Difference of Arrival Localization Study for SAR Systems over LoRaWAN. *Procedia Computer Science*. 2020. Вип. 175. С. 292–299.

7. Mesh network topology. URL: https://www.techtarget.com/iotagenda/definition/mesh-network-topology-mesh-network (дата звернення: 16.11.2022).

8. How Wireless Mesh Networks Work. URL: https://computer.howstuffworks.com/how-wireless-mesh-networks-work.htm (дата звернення: 18.11.2022).

9. Arduino Nano. URL: https://store.arduino.cc/products/arduino-nano (дата звернення: 04.12.2022).

10. Arduino Nano V3.0 AVR ATmega328P. URL: https://ardushop.in.ua/arduino/arduino-nano-v3-avr-atmega328 (дата звернення: 05.12.2022).

11. ESP32 Series. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 06.12.2022).

12. ESP32. URL: https://www.espressif.com/en/products/socs/esp32 (дата звернення: 06.12.2022).

13. Getting Started with ESP32 | Introduction to ESP32. URL: https://www.electronicshub.org/getting-started-with-esp32 (дата звернення: 06.12.2022).

14. Raspberry Pi 4 Model B. URL: https://www.amazon.com/Raspberry-Model-2019-Quad-Bluetooth/dp/B07TC2BK1X?th=1 (дата звернення: 07.12.2022).

15. What is the Raspberry Pi 4? Everything you need to know about the tiny, low-cost computer. URL: https://www.zdnet.com/article/what-is-the-raspberry-pi-4-everything-you-need-to-know-about-the-tiny-low-cost-computer (дата звернення: 08.12.2022).

16. Raspberry Pi 4 Tech Specs. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications (дата звернення: 08.12.2022).

17. Getting Started with the Raspberry Pi Zero 2 W. URL: https://www.makeuseof.com/how-to-set-up-raspberry-pi-zero-2-w (дата звернення: 09.12.2022).

18. Мікрокомп'ютер Raspberry Pi Zero 2 W. URL: https://evo.net.ua/mikrokomputer-raspberry-pi-zero-2-w (дата звернення: 10.12.2022).

19. Ultrasonic Distance Sensor (HC-SR04). URL: https://www.piborg.org/sensors-1136/hc-sr04 (дата звернення: 13.12.2022).

20. JSN - SR04T. URL: https://www.tinytronics.nl/shop/en/sensors/distance/waterproof-ultrasonic-sensor-jsn-sr04t (дата звернення: 14.12.2022).

21. Me Ultrasonic Sensor. URL: http://docs.makeblock.com/diy-platform/en/electronic-modules/sensors/me-ultrasonic-sensor.html (дата звернення: 16.12.2022).

22. Xiaomi Mesh System AX3000 1 pack. URL: https://allo.ua/ua/wi-fi-routery/xiaomi-mesh-system-ax3000-1-pack-dvb4315gl.html (дата звернення: 31.12.2022).

23. HTTPS. URL: https://en.wikipedia.org/wiki/HTTPS (дата звернення: 04.01.2023).

24. HAProxy. URL: https://www.haproxy.org/#desc (дата звернення: 04.01.2023).

25. Elixir. URL: https://elixir-lang.org (дата звернення: 05.01.2023).

26. Flutter. Build apps for any screen. URL: https://flutter.dev (дата звернення: 06.01.2023).

27. How secure is WebSocket? URL: https://blog.passwork.pro/how-secure-is-websocket (дата звернення: 06.01.2023).

28. Rust. A language empowering everyone to build reliable and efficient software. URL: https://www.rust-lang.org (дата звернення: 07.01.2023).

29. InfluxDB Clustering - High Availability and Scalability. URL: https://www.influxdata.com/blog/influxdb-clustering (дата звернення: 07.01.2023).

30. Плюснін К. А., Пузирьов С. В. Визначення джерела звуку за допомогою mesh-мережі. Могилянські читання – 2022 : тези доп. XXV Всеукр. наук.-метод. конф. Миколаїв, 7–11 листоп. 2022 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2022. С. 76–77.

ДОДАТОК А

Концептуальна схема апаратно-програмного комплексу

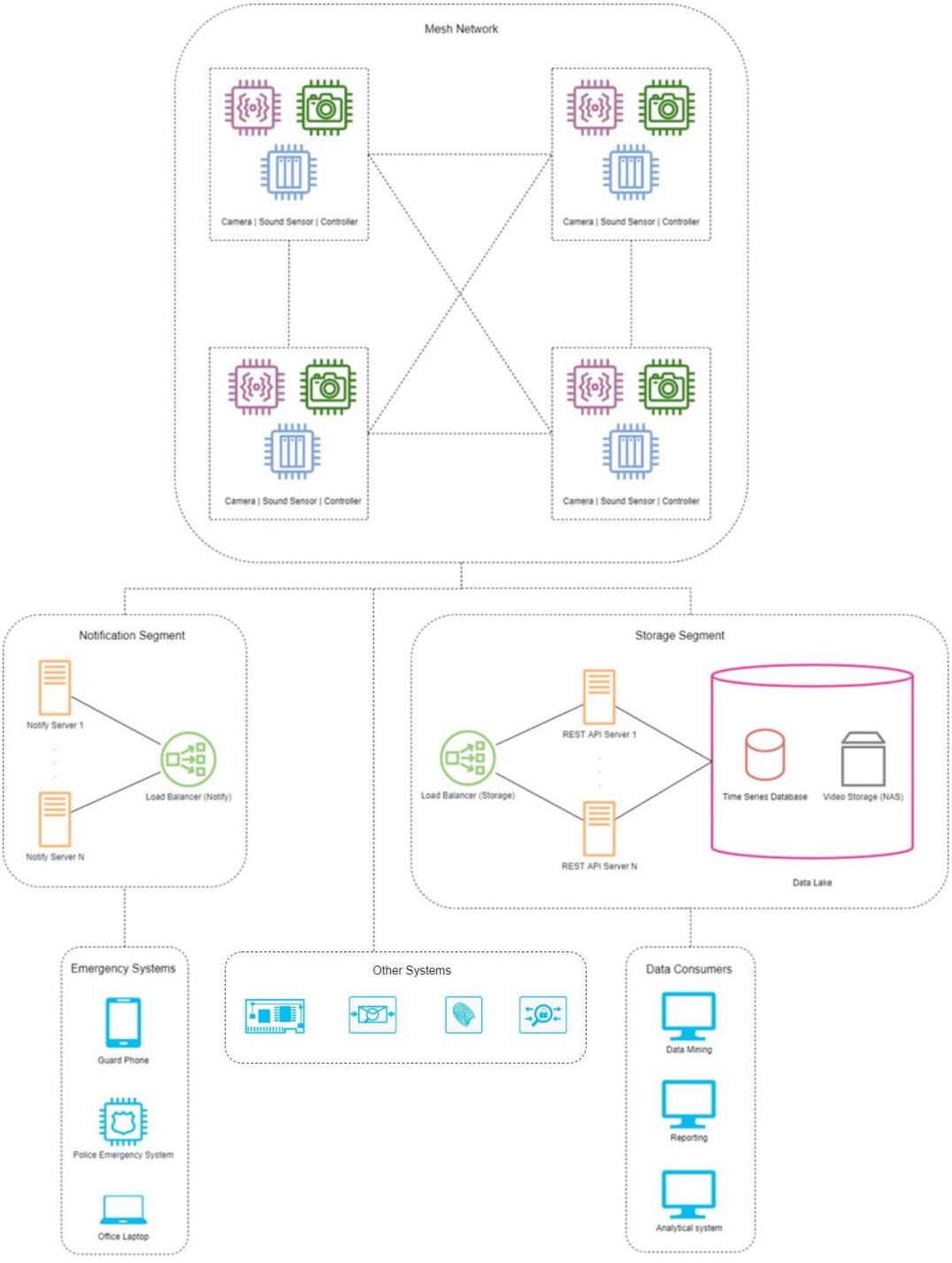


Рисунок А.1 – Концептуальна схема апаратно-програмного комплексу

ДОДАТОК Б

Структурна діаграма апаратно-програмного комплексу

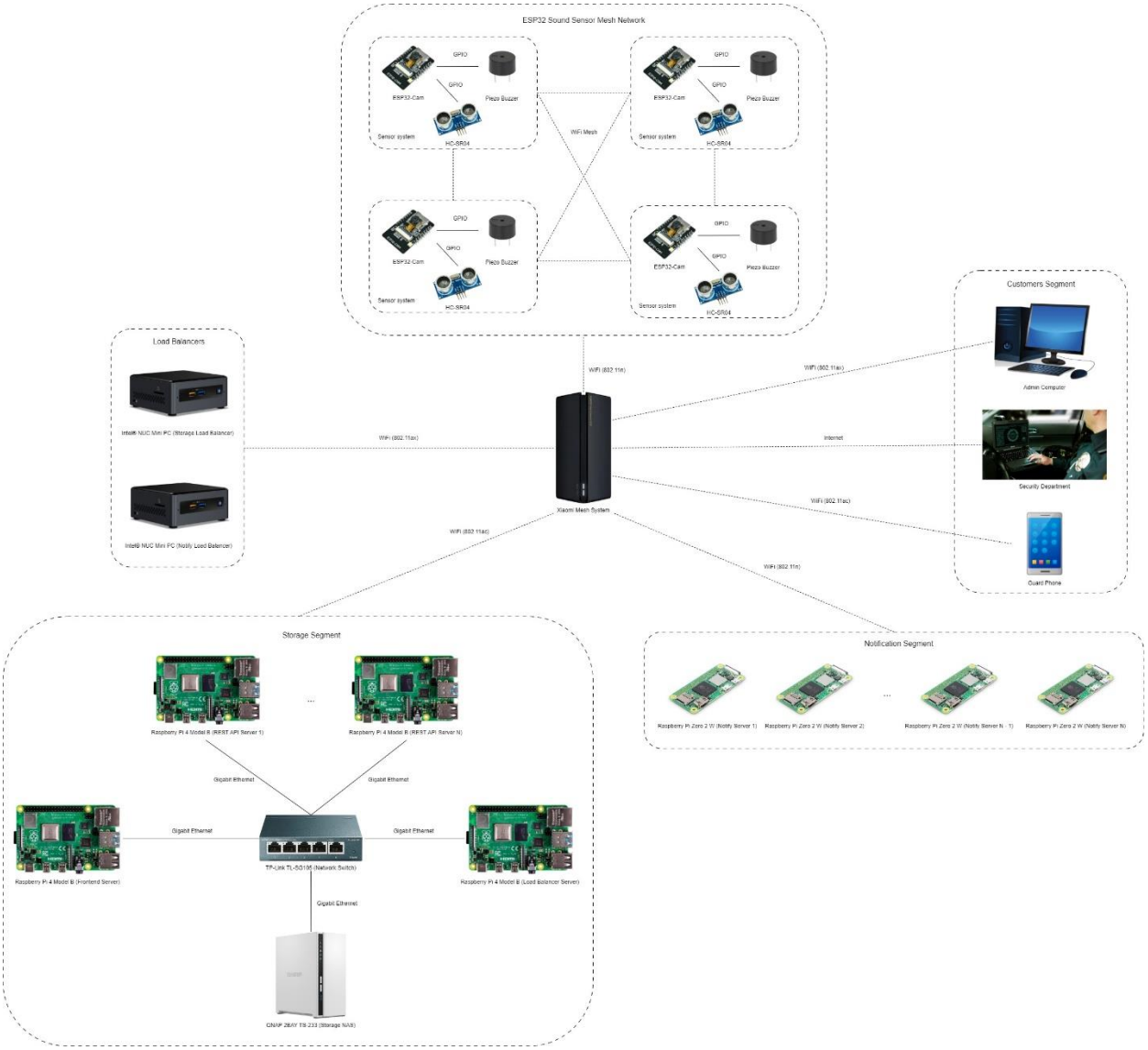


Рисунок Б.1 – Діаграма апаратної архітектури апаратно-програмного комплексу

ДОДАТОК В

Діаграма програмної архітектури апаратно-програмного комплексу

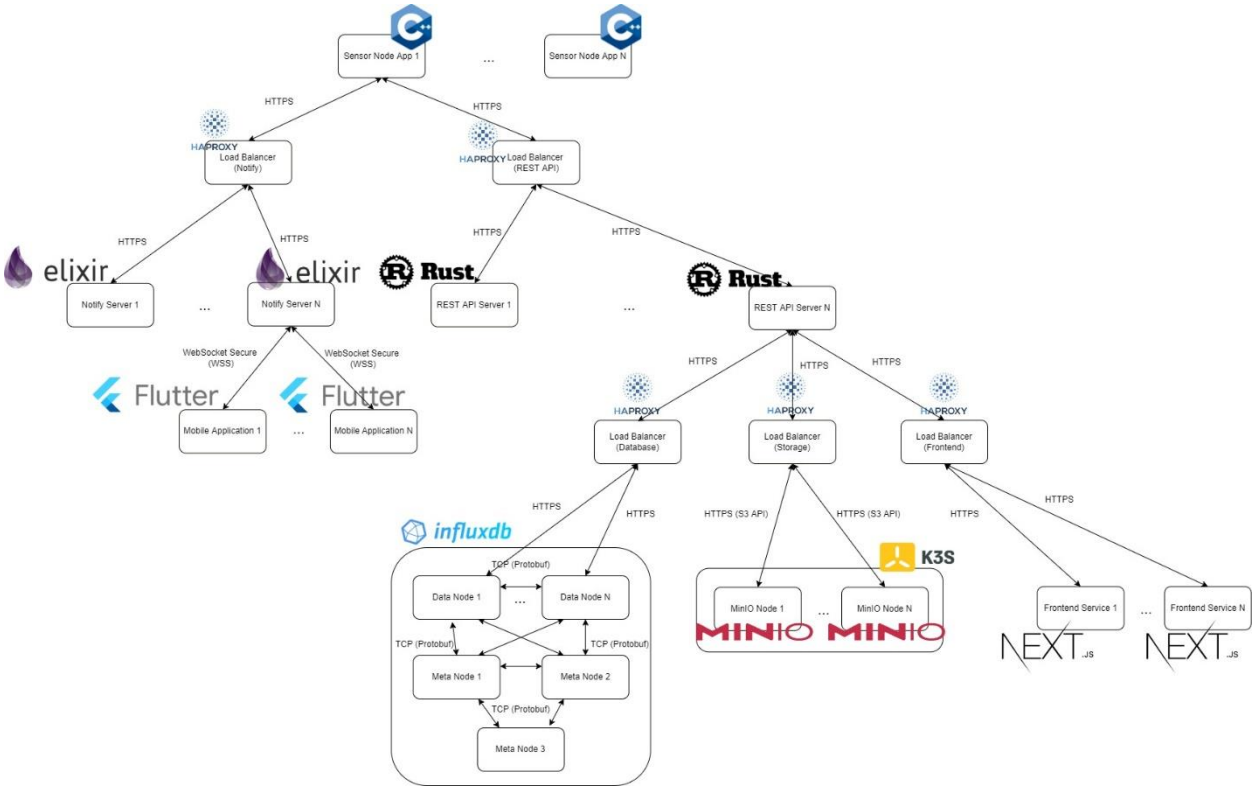


Рисунок В.1 – Діаграма програмної архітектури апаратно-програмного комплексу

ДОДАТОК Г

Програмний код апаратно-програмного комплексу

```
#include "esp_camera.h"
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "driver/rtc_io.h"
#include <ESPAsyncWebServer.h>
#include <StringArray.h>
#include <SPIFFS.h>
#include <FS.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>

const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

#define FILE_PHOTO "/photo.jpg"
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#define OLED_WIDTH 128
#define OLED_HEIGHT 64
#define OLED_ADDR   0x3C

Adafruit_SSD1306 display(OLED_WIDTH, OLED_HEIGHT);

const char* serverName = "http://192.168.1.106:1880/update";
const int trigPin = 12;
const int echoPin = 15;

#define SOUND_SPEED 0.034

long duration;
float distance;

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

void setup() {
  Serial.begin(115200);
  display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR);
  display.clearDisplay();
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

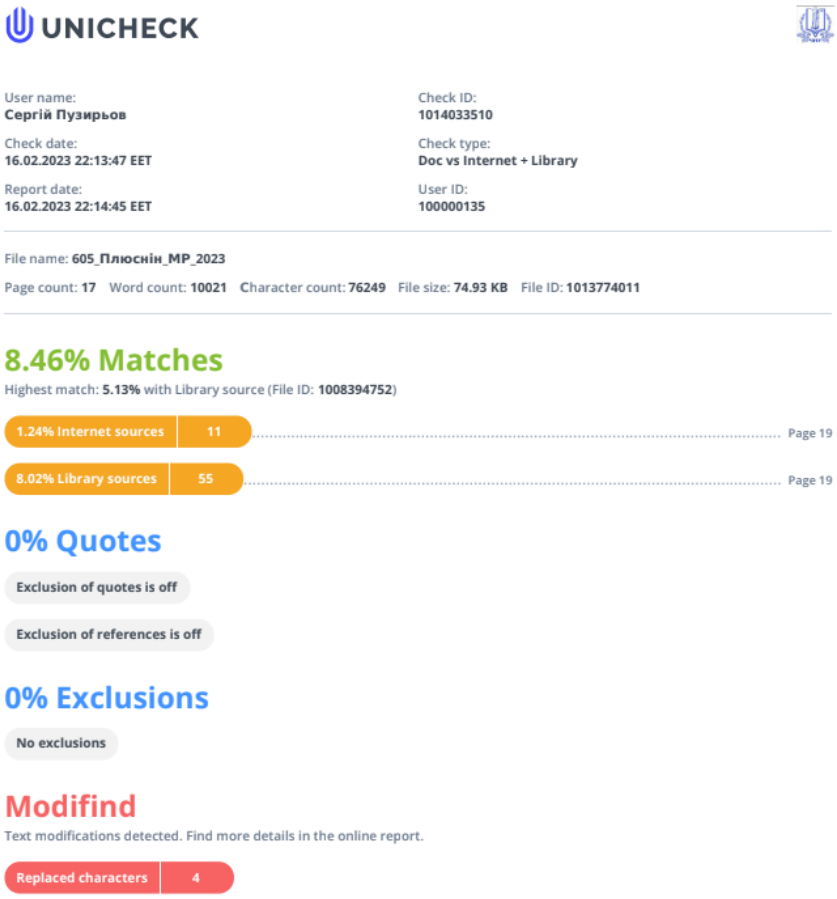
```
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  if (!SPIFFS.begin(true)) {
    Serial.println("An Error has occurred while mounting SPIFFS");
    ESP.restart();
  }
  else {
    delay(500);
    Serial.println("SPIFFS mounted successfully");
  }
  Serial.println("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  if (psramFound()) {
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    ESP.restart();
  }
  timeClient.begin();
}

void loop() {
  timeClient.update();
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * SOUND_SPEED/2;

  if (distance > 0) {
    if(WiFi.status()== WL_CONNECTED){
      date = timeClient.getFormattedTime()
      capturePhotoSaveSpiffs();
      printInfo(distance, date);
      WiFiClient client;
      HTTPClient http;
```

```
      http.begin(client, serverName);
      File photo = SPIFFS.open(FILE_PHOTO, "r");
      if (!photo) {
        Serial.println("Failed to open file for reading");
        return;
      }
      client.write(photo);
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
      String                              httpRequestData                                =
"distance="+distance+"&sensor_number="+sensor_number+"&datetime="+date;
      int httpResponseCode = http.POST(httpRequestData);
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      http.end();
    }
    else {
      Serial.println("WiFi Disconnected");
    }
  }
  delay(1000);
}

bool checkPhoto( fs::FS &fs ) {
  File f_pic = fs.open( FILE_PHOTO );
  unsigned int pic_sz = f_pic.size();
  return ( pic_sz > 100 );
}

void capturePhotoSaveSpiffs( void ) {
  camera_fb_t * fb = NULL; // pointer
  bool ok = 0;
  do {
    Serial.println("Taking a photo...");
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      return;
    }
    Serial.printf("Picture file name: %s\n", FILE_PHOTO);
    File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);
    if (!file) {
      Serial.println("Failed to open file in writing mode");
    }
    else {
      file.write(fb->buf, fb->len);
      Serial.print("The picture has been saved in ");
      Serial.print(FILE_PHOTO);
      Serial.print(" - Size: ");
      Serial.print(file.size());
      Serial.println(" bytes");
    }
    file.close();
    esp_camera_fb_return(fb);

    ok = checkPhoto(SPIFFS);
  } while ( !ok );
}

void printInfo(double distance, String date) {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println(date);
  display.println(distance);
  display.display();
  delay(5000);
}
```

ДОДАТОК Ґ
ДОВІДКА

про перевірку на унікальність пояснювальної записки
кваліфікаційної магістерської роботи
на тему: «Система виявлення джерел звуку на базі mesh-мережі»
студента спеціальності 123 «Комп'ютерна інженерія»,
групи 605м
Плюснін Костянтин Андрійович
прізвище, ім'я, по-батькові

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck.Результат перевірки тексту роботи: схожість складає 8,46%.


UNICHECK

User name: Сергій Пузирьов
Check ID: 1014033510
Check date: 16.02.2023 22:13:47 EET
Check type: Doc vs Internet + Library
Report date: 16.02.2023 22:14:45 EET
User ID: 100000135

File name: 605_Плюснін_MP_2023
Page count: 17 Word count: 10021 Character count: 76249 File size: 74.93 KB File ID: 1013774011

8.46% Matches
Highest match: 5.13% with Library source (File ID: 1008394752)
1.24% Internet sources 11 Page 19
8.02% Library sources 55 Page 19

0% Quotes
Exclusion of quotes is off
Exclusion of references is off

0% Exclusions
No exclusions

Modifind
Text modifications detected. Find more details in the online report.
Replaced characters 4

Студент

_____К. А. Плюснін
підпис ініціали, прізвище

Керівник
канд. фіз.-мат. наук, доцент
_____С. В. Пузирьов
підпис ініціали, прізвище

Дата: «___»_____2023 р.