Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютеризована система голосового інформування людей з вадами зору на базі Raspberry PI*

Виконав: студент 4 курсу, групи СІ-41
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

	(підпис)	*Конфедерат О.О.* (прізвище та ініціали)
Керівник	(підпис)	*Стадник Н.Б.* (прізвище та ініціали)
Нормоконтроль	(підпис)	*Луцик Н.С.* (прізвище та ініціали)
Завідувач кафедри	(підпис)	*Осухівська Г.М.* (прізвище та ініціали)
Рецензент	(підпис)	*Марценко С.В.* (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
«27» січня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студента *Конфедерат Олександри Олександрівни*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система голосового інформування людей з вадами зору на базі Raspberry PI*

Керівник роботи *Стадник Наталія Богданівна, к.т.н., старший викладач кафедри КС*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від *« 27 » січня 2025 року № 4/7-53*
2. Термін подання студентом завершеної роботи 19.06.2025 р.
3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз технічного завдання
2. Проєктна частина
3. Практична частина
4. Безпека життєдіяльності, основи охорони праці.
Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Схема структурна
2. Схема електрична принципова
3. Блок-схема алгоритму роботи
4. Блок-схема алгоритму програмного забезпечення

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*дтн.,професор Пилипець М.І.*		

7. Дата видачі завдання *27.01.2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка технічного завдання*	*27.01. – 02.02.*	*Викон.*
2.	*Аналіз технічного завдання*	*03.02. – 05.03.*	*Викон.*
3.	*Аналіз технологій розробки системи голосового інформування*	*06.03. – 25.03.*	*Викон.*
4.	*Обґрунтування вибору апаратного забезпечення*	*26.03. – 16.04.*	*Викон.*
5.	*Реалізація програмного забезпечення системи*	*17.04. – 10.05.*	*Викон.*
6.	*Реалізація апаратного забезпечення системи*	*11.05.– 01.06.*	*Викон.*
7.	*Безпека життєдіяльності, основи охорони праці*	*2.06. – 13.06.*	*Викон.*
8.	*Оформлення пояснювальної записки і графічного матеріалу*	*02.06. – 22.02.*	*Викон.*
9.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*9.06. – 15.06.*	*Викон.*
10.	*Попередній захист кваліфікаційної роботи бакалавра*	*16.06.– 22.06.*	*Викон.*
11.	*Захист кваліфікаційної роботи бакалавра*	*26.06.*	

Студент _____ *Конфедерат О.О.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Стадник Н.Б.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Конфедерат О.О. Комп'ютеризована система голосового інформування людей з вадами зору на базі Raspberry PI: робота на здобуття кваліфікаційного ступеня бакалавра: спец. 123 — комп’ютерна інженерія. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2025.

Ключові слова: Raspberry Pi, асистивні технології, люди з вадами зору, комп'ютерний зір, ультразвуковий сенсор, синтез мовлення, розпізнавання перешкод, вбудована система.

Кваліфікаційна робота присвячена розробці комп’ютеризованої системи голосового інформування для людей з вадами зору, яка здатна в реальному часі виявляти перешкоди та повідомляти про них користувача. Пояснювальна записка складається з чотирьох розділів.

У першому розділі проведено аналіз вимог до проектованої системи, а також існуючих асистивних технологій. Обґрунтовано актуальність теми та визначено основні завдання дослідження.

У другому розділі було розроблено архітектуру комп’ютеризованої системи. Обґрунтовано вибір апаратної складової на базі Raspberry Pi, сенсорних модулів та програмних засобів.

У третьому розділі описано програмну реалізацію функцій системи мовою Python та процес збирання апаратного прототипу. Також проведено тестування працездатності системи на ефективність виявлення перешкод.

В четвертому розділі розглянуті питання безпеки життєдіяльності та охорони праці при роботі з комп’ютерною та електронною технікою.

ANNOTATION

Konfederat O.O. Computerized voice information system for visually impaired people based on Raspberry PI: Bachelor’s Graduation Thesis: speciality 123 — computer engineering. Ternopil: Ternopil Ivan Puluj National Technical University, 2025.

Keywords: Raspberry Pi, assistive technologies, visually impaired people, computer vision, ultrasonic sensor, speech synthesis, obstacle detection, embedded system.

The bachelor's thesis is devoted to the development of a computerized voice notification system for visually impaired individuals, capable of detecting obstacles in real time and informing the user about them. The explanatory note consists of four chapters.

The first chapter presents an analysis of the requirements for the designed system, as well as a review of existing assistive technologies. The relevance of the topic is substantiated, and the main research objectives are defined.

The second chapter outlines the architecture of the computerized system. The choice of hardware components based on Raspberry Pi, sensor modules, and software tools is justified.

The third chapter describes the software implementation of the system’s functions in Python and the process of assembling the hardware prototype. System testing was also conducted to evaluate the effectiveness of obstacle detection.

The fourth chapter addresses life safety and occupational health issues when working with computer and electronic equipment.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

AI – Artificial Intelligence

API – Application Programming Interface

CPU – Central Processing Unit

CSI – Camera Serial Interface

GPIO – General-Purpose Input/Output

GPU – Graphics Processing Unit

HDMI – High Definition Multimedia Interface

IoT – Internet of Things

IP – Internet Protocol

ISP – Image Signal Processor

OS – Operating System

OpenCV – Open Source Computer Vision Library

USB – Universal Serial Bus

Wi-Fi – Wireless Fidelity

БЖД – Безпека життєдіяльності

ДСАНПІН – Державні санітарні норми і правила

ЕМП – Електромагнітне поле

ЕСКД – Єдина система конструкторської документації

КРБ – Кваліфікаційна робота бакалавра

КС – Комп'ютеризована система

КСГІ – Комп'ютеризована система голосового інформування

ОП – Охорона праці

ОС – Операційна система

ПЗ – Програмне забезпечення

ВСТУП

Стрімкий розвиток сучасного суспільства висувають нові вимоги до створення інклюзивного та доступного середовища для всіх людей. Однією з ключових перепон на цьому шляху є труднощі з орієнтацією у просторі, з якими стикаються особи з вадами зору. Тому розробка інноваційних асистивних технологій, що сприяють їхній безпеці та самостійності, є пріоритетним завданням для комп'ютерної інженерії.

У кваліфікаційній роботі досліджується проектування та створення комп'ютеризованої системи голосового інформування, призначеної для людей з порушеннями зору. В основі системи лежить використання сучасної мікрокомп'ютерної техніки та сенсорних технологій, що дозволяє в режимі реального часу аналізувати навколишнє середовище.

Метою кваліфікаційної роботи є розробка портативної та ефективної системи, яка дозволить підвищити рівень безпеки та мобільності людей з вадами зору шляхом своєчасного виявлення перешкод. Реалізація цієї мети сприятиме глибшій соціальній інтеграції та зростанню незалежності користувачів, що є важливою суспільною задачею в контексті побудови безбар'єрного простору.

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз наявних асистивних технологій та пристроїв для навігації;
- спроектувати архітектуру та електричну схему портативного пристрою;
- розробити алгоритм обробки сенсорних даних та відповідне програмне забезпечення;
- зібрати та провести тестування робочого прототипу системи.

Створення комп'ютеризованої системи голосового інформування має великий потенціал для покращення якості життя людей з вадами зору. Впровадження таких технологій може стати значущим кроком на шляху до

формування більш справедливого та інклюзивного суспільства, що є критично важливим у сучасних гуманістичних реаліях.

Застосування розробленої системи дозволить не лише запобігати зіткненням з перешкодами, але й підвищить впевненість користувачів у власних силах, що є ключовим фактором для їхньої успішної соціальної адаптації та особистої свободи.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Основні технічні вимоги до комп’ютеризованої системи голосового інформування

На основі аналізу поставленої задачі було сформовано комплекс технічних вимог до комп’ютеризованої системи голосового інформування (КСГІ), що детально викладені у технічному завданні. Ключові вимоги до системи охоплюють її функціональність, показники продуктивності та конструктивні особливості.

Система повинна реалізовувати гібридний підхід до аналізу оточення, поєднуючи дані з оптичних та ультразвукових сенсорів. Основним завданням є виявлення статичних/динамічних перешкод та їх класифікація за допомогою нейромережевих моделей комп'ютерного зору [5]. Результати аналізу доводяться до користувача через ієрархічну трьохрівневу систему зворотного зв'язку: інформаційні голосові сповіщення, попереджувальні звукові сигнали та критичні тактильно-звукові сигнали тривоги. Для забезпечення надійності передбачено режим відмовостійкості, при якому відмова модуля камери не призводить до повної зупинки системи, а переводить її в базовий режим виявлення перешкод за допомогою ультразвукового давача;

Ключові показники продуктивності визначають ефективність системи в реальних умовах. Робочий діапазон виявлення перешкод ультразвуковим давачем HC-SR04 встановлено в межах від 0.2 до 4 метрів [3], тоді як ефективна дистанція для розпізнавання об'єктів камерою складає 1-5 метрів. Латентність системи є критичним параметром: час реакції на небезпеку на критичній відстані (спрацювання ультразвукового давача) не повинен перевищувати 250 мс, а загальна затримка для повного циклу "розпізнавання-

сповіщення" — 1 секунди. Час автономної роботи пристрою від портативного джерела живлення має становити не менше 8 годин безперервного використання.

Пристрій проектується як компактний та легкий носимий пристрій з габаритами, що не перевищують 80×60×40 мм, та вагою до 200 г. Інтерфейс взаємодії з користувачем (HMI) має бути мінімалістичним та базуватися на тактильно розрізнюваних органах керування для можливості використання "наосліп". Конструкція повинна бути надійною та стійкою до незначних механічних впливів. Програмний код, нейромережева модель та файли налаштувань зберігаються локально на MicroSD-карті, що забезпечує повну автономність системи та її незалежність від зовнішніх серверів чи доступу до мережі Інтернет.

1.2 Аналіз існуючих рішень

1.2.1 Методи орієнтації для людей з вадами зору

У світовій практиці для допомоги людям з вадами зору використовуються два основні підходи: традиційний та електронно-технологічний.

Традиційні методи – це біла тростина та собака-поводир. Біла тростина є простим тактильним інструментом для «промацування» шляху, що дозволяє виявляти низькі перешкоди та зміни рельєфу. Її головні недоліки – обмежений радіус дії та нездатність попереджати про загрози на рівні вище пояса. Собака-поводир – це високоінтелектуальний помічник, здатний до комплексного аналізу ситуації, проте його підготовка та утримання є дуже дорогими і вимагають від власника великої відповідальності. Це створює очевидну потребу в доступних технологічних рішеннях, які б доповнювали або замінювали традиційні засоби [16].

Традиційні методи орієнтації для людей з вадами зору представлені на рисунку 1.1.

а) б)

Рисунок 1.1 – Традиційні методи орієнтації: а) – використання білої тростини; б) – собака-поводир

Електронні методи є технологічною відповіддю на обмеження традиційних засобів. Вони базуються на використанні сенсорів (ультразвукових, інфрачервоних) та камер для сканування простору. Отримані дані обробляються мікропроцесорним модулем, який генерує для користувача попереджувальні сигнали – звук, вібрацію або голос. Це дозволяє виявляти перешкоди на різних відстанях та висотах [6].

1.2.2 Існуючі електронні системи допомоги

Ринок електронних асистентів містить як комерційні продукти, так і академічні чи аматорські розробки.

Аматорські та академічні проєкти часто створюються на базі популярних платформ, таких як Arduino, ESP32 та Raspberry Pi. Проєкти на Arduino зазвичай реалізують базовий функціонал з ультразвуковим давачем та звуковим/вібраційним сигналом. Платформа ESP32 дозволяє додавати зв’язок зі смартфоном через Bluetooth. Найбільш функціональні системи розробляються на Raspberry Pi, що дає змогу використовувати камери та алгоритми комп’ютерного зору для розпізнавання об’єктів. Спільним недоліком таких розробок часто є недостатня ергономіка та потреба в доопрацюванні.

Професійні комерційні системи пропонують готове та надійне рішення. Найбільш відомі з них показано на рисунку 1.2.

а) б) в)

Рисунок 1.2 – Професійні системи допомоги: а) – WeWalk Smart Cane; б) – OrCam MyEye; в) – Sunu Band

На рисунку 1.2 (а) представлено WeWalk Smart Cane. Це гібридний пристрій, що модернізує класичну білу тростину, не замінюючи її, а розширюючи її можливості. Основний модуль кріпиться до верхньої частини будь-якої стандартної тростини. Він оснащений вбудованим ультразвуковим сенсором, який сканує простір попереду користувача на рівні грудей до голови. При виявленні перешкод, таких як гілки дерев, дорожні знаки або інші об’єкти, які неможливо відчути звичайною тростиною, пристрій миттєво попереджає користувача за допомогою вібрації на руків’ї. Ключовою перевагою WeWalk є його здатність до інтеграції зі смартфоном через Bluetooth. За допомогою мобільного додатку користувач може отримувати покрокові навігаційні інструкції, інформацію про зупинки громадського транспорту та інші дані, що значно спрощує орієнтацію в міському середовищі.

На рисунку 1.2 (б) зображено систему OrCam MyEye. Цей пристрій належить до класу технологій «штучного зору» і являє собою мініатюрну камеру, яка магнітним кріпленням фіксується на дужці окулярів користувача. Його головна мета – не навігація, а інтерпретація візуальної інформації. Завдяки вбудованому процесору та алгоритмам штучного інтелекту, OrCam

MyEye може в реальному часі «читати» будь-який друкований або цифровий текст (меню в ресторані, газетні статті, вивіски), розпізнавати обличчя людей із попередньо створеної бази, ідентифікувати номінали банкнот, кольори та навіть конкретні товари в магазині. Вся інформація передається користувачеві через мініатюрний динамік у вигляді чітких аудіоповідомлень. Це надзвичайно потужне рішення, що кардинально підвищує рівень незалежності у повсякденному житті, однак його висока вартість робить його недоступним для широкого кола користувачів.

На рисунку 1.2 (в) показано браслет-сонар Sunu Band. Цей пристрій є компактним носимим сонаром, призначення для покращення просторової обізнаності та захисту верхньої частини тіла. Браслет, що носять на зап’ясті, використовує технологію ехолокації: він випромінює високочастотні ультразвукові хвилі та аналізує їх відбиття від об’єктів. Відстань до перешкоди в напрямку руки користувача перетворюється на тактильний зворотний зв’язок – ритмічну вібрацію. Чим ближче об’єкт, тим інтенсивнішою та частішою стає вібрація. Sunu Band є ефективним доповненням до білої тростини чи собаки-поводиря, оскільки дозволяє «відчувати» простір навколо себе на відстані до 5 метрів, уникати зіткнень з іншими людьми в натовпі та впевненіше проходити через дверні прорізи. Його переваги – це вільні руки та непомітність у використанні.

1.3 Обґрунтування вибору компонентів для розробки

Процес вибору компонентів для розробки базувався на мультикритеріальному аналізі, де основними вимогами виступали: достатня обчислювальна потужність для задач комп'ютерного зору, низьке енергоспоживання, наявність необхідних інтерфейсів та доступна вартість.

В якості обчислювального ядра системи, з урахуванням вимоги до обробки відео в реальному часі та запуску нейромережевих моделей, було обрано одноплатний комп'ютер Raspberry Pi 3 Model B [9]. У порівнянні з

більш потужними альтернативами, як-от NVIDIA Jetson Nano, Raspberry Pi 3 пропонує оптимальний компроміс між продуктивністю та ціною. Його чотирьохядерний процесор ARM Cortex-A53 здатний виконувати інференс оптимізованих моделей TensorFlow Lite [2], а наявність повноцінної ОС Raspberry Pi OS та величезної спільноти значно спрощує розробку та налагодження програмного забезпечення.

Для забезпечення надійності аналізу оточення було прийнято рішення про реалізацію гібридної сенсорної системи. Використання лише одного типу давача є недостатнім: камера неефективна в темряві та не може точно виміряти відстань, тоді як ультразвуковий давач не здатний ідентифікувати об'єкти. Для модуля комп'ютерного зору було обрано модель Waveshare RPi Camera (G), вирішальним фактором для якої став її ширококутний об'єктив («риб'яче око»), що забезпечує поле зору понад 160°. Це є суттєвою перевагою над стандартними камерами, оскільки дозволяє системі виявляти бічні загрози. В якості давача відстані було обрано ультразвуковий модуль HC-SR04 [3], який, на відміну від інфрачервоних давачів, стабільно працює за будь-яких умов освітлення та не залежить від кольору об'єкта.

Для інформування користувача спроектовано трьохрівневу систему сповіщень. Активні динаміки з живленням від USB були обрані для голосових повідомлень через простоту інтеграції, оскільки вони не вимагають зовнішнього підсилювача. Для сигналів тривоги було обрано стандартні та енергоефективні активний зумер та вібромотор, що забезпечують надійний звуковий та тактильний відгук.

В якості системи живлення було обрано готовий портативний акумулятор (Power Bank) замість збирання власного модуля. Це рішення обґрунтоване значно вищою надійністю та безпекою, оскільки комерційні пристрої мають вбудовані контролери захисту від перезаряду, короткого замикання та перегріву, що є критично важливим для носимого пристрою.

РОЗДІЛ 2 ПРОЄКТНА ЧАСТИНА

2.1 Розробка структури системи голосового інформування

Для забезпечення надійності, модульності та можливості подальшої модернізації, структура комп'ютеризованої системи голосового інформування (КСГІ) розроблена за ієрархічним принципом [25]. Вона складається з чотирьох логічно завершених підсистем, кожна з яких виконує свій набір функцій.

Структуру системи візуально представлено на блок-схемі (рис. 2.1).

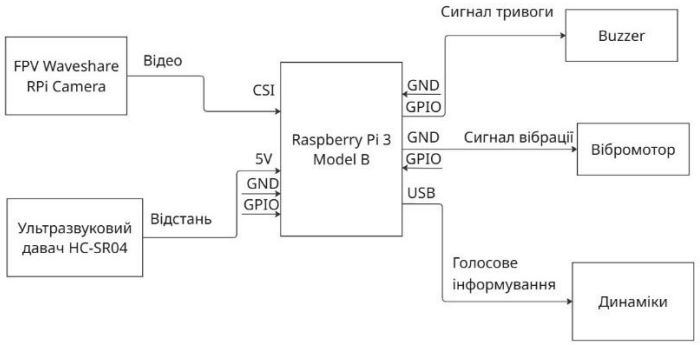


Рисунок 2.1 – Структура системи голосового інформування

Raspberry Pi виступає як центральний обчислювальний вузол, що аналізує дані з сенсорів та генерує голосові сповіщення. Система є автономною та не потребує підключення до ПК у процесі роботи.

Система поєднує дві ключові технології для сприйняття оточення: комп'ютерний зір для розпізнавання об'єктів [5] та ультразвукову локацію для вимірювання відстані [3].

Комп'ютерний зір реалізується за допомогою камери Waveshare RPi Camera (G) та нейромережевих алгоритмів. Камера захоплює зображення, яке перетворюється на цифровий масив даних. Цей масив подається на вхід

попередньо навченої нейронної мережі, яка аналізує його та видає результат у вигляді назви об’єкта та його координат на зображенні [12].

Ультразвукова локація працює за принципом ехолота. Мікрокомп’ютер подає короткий імпульс на Trig-пін давача HC-SR04, який випромінює пачку ультразвукових хвиль. Хвилі відбиваються від перешкоди і повертаються на Echo-пін давача. Вимірявши час між відправкою та отриманням сигналу, система обчислює точну відстань до перешкоди [3].

Raspberry Pi обробляє дані з обох джерел. Унікальний ідентифікатор об’єкта («людина», «сходи») з камери та дані про відстань від ультразвукового давача поєднуються для формування змістовного голосового повідомлення.

Загальна схема з’єднання Raspberry PI з основними модулями представлена на рисунку 2.2.

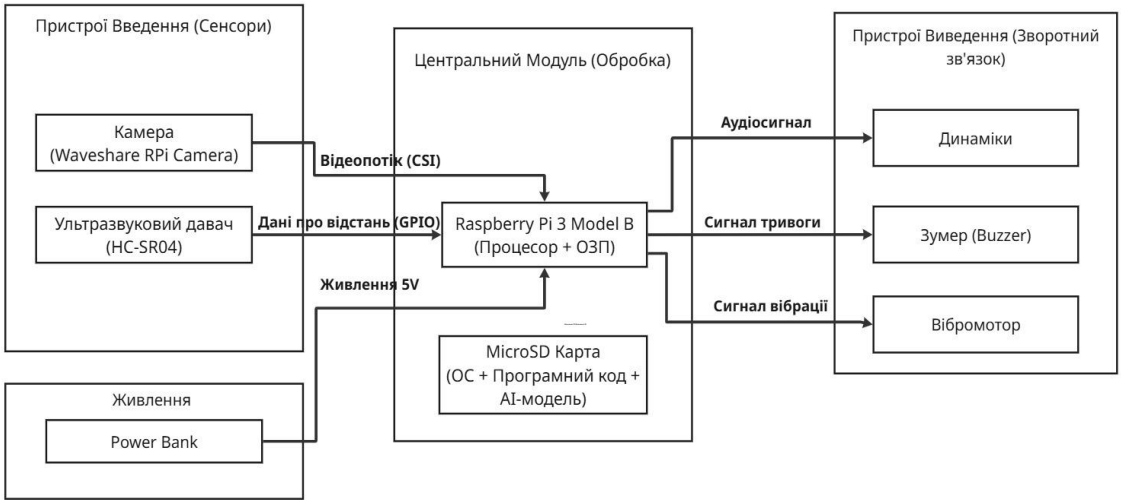


Рисунок 2.2 – Загальна схема апаратних з’єднань

Підключення всієї периферії до Raspberry Pi здійснюється через 40-контактну гребінку GPIO, яка надає доступ як до ліній живлення, так і до програмованих контактів для передачі керуючих сигналів та отримання даних.

Схема з’єднань компонентів, що підключаються через піни на гребінці GPIO зображена на рисунку 2.3.

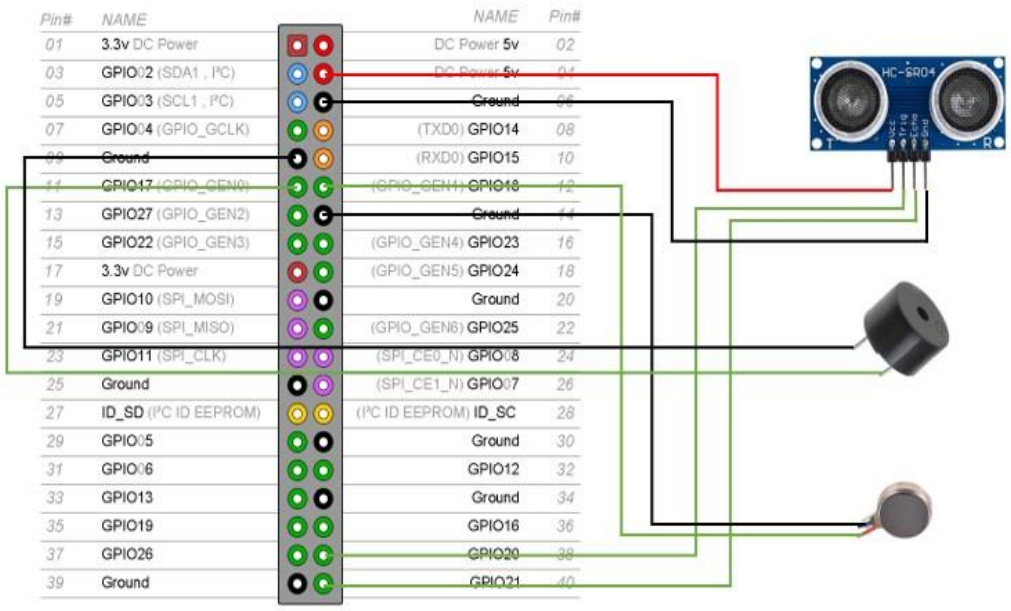


Рисунок 2.3 – Схема з’єднань периферійних модулів

Для ультразвукового давача використовуються лише 4 контакти Raspberry PI. Два піни, що використовуються для живлення (5V та GND), а два – для передачі сигналів (Trig та Echo), що забезпечує повний цикл роботи давача: команда від Raspberry Pi, вимірювання давачем і передача результату назад на Raspberry Pi для обчислень.

Таблиця 2.1 – Підключення модуля HC-SR04 до Raspberry Pi

Виводи модуля HC-SR04	Виводи Raspberry PI
VCC (Живлення)	Pin 4 (5V)
Trig (Сигнал запуску)	Pin 38 (GPIO20)
Echo (Сигнал відповіді)	Pin 40 (GPIO 21)
GND (Земля)	Pin 6 (GND)

Для підключення зумера (Buzzer) використовуються два піни. Підключення реалізує просте електронне коло, яким керує Raspberry Pi, зумер додає до системи ще один рівень сповіщень.

Таблиця 2.2 – Підключення зумера (Buzzer) до Raspberry Pi

Виводи зумера	Виводи Raspberry PI
I/P	Pin 11 (GPIO17)
GND (Земля)	Pin 9 (GND)

Вібромотор використовується для тактильного сповіщення користувача про критичну небезпеку. Для його підключення використовуються два піни і в результаті створюється вібрація.

Таблиця 2.3 – Підключення вібромотора до Raspberry Pi

Виводи вібромотора	Виводи Raspberry PI
I/P	Pin 12 (GPIO18)
GND (Земля)	Pin 14 (GND)

Підключення модуля камери Waveshare RPi Camera (G) до Raspberry Pi Model B підключається до спеціалізованого інтерфейсу CSI (Camera Serial Interface) [9]. Цей інтерфейс призначений для прямого з'єднання цифрових камер з процесорами у вбудованих системах (рис. 2.4).

Рисунок 2.4 – Підключення камери до CSI

Аудіосистема підключається до Raspberry Pi за допомогою двох окремих кабелів: 3.5 мм аудіокабелю (AUX) та живлення через USB. AUX передає

аналоговий лінійний сигнал, згенерований програмним синтезатором мовлення.

Моделювання з’єднань елементів системи голосового інформування представлено на рисунку 2.5.

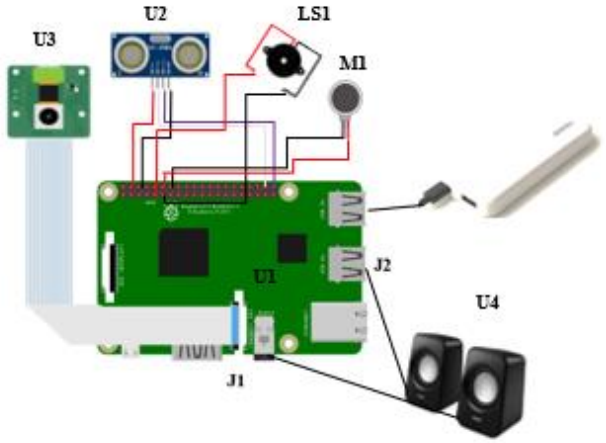


Рисунок 2.5 – Моделювання з’єднань системи

Для демонстрації взаємозв'язку всіх апаратних модулів та узагальнення інформації з таблиць 2.1-2.3, було розроблено загальну схему електричну принципову системи, наведену на рисунку 2.6.

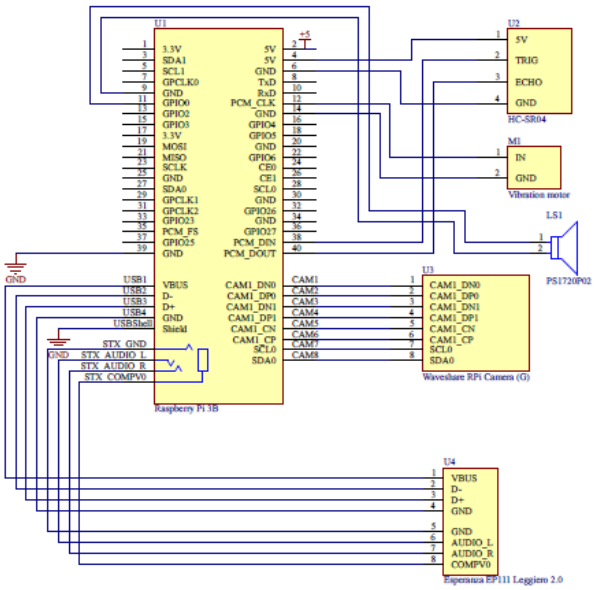


Рисунок 2.6 - Схема електрична принципова з’єднань системи

2.2 Обґрунтування вибору апаратного забезпечення системи

Вибір апаратної платформи та периферійних модулів є фундаментальним етапом проєктування, що безпосередньо визначає функціональні можливості, продуктивність та надійність розробленої системи. Компоненти підбирались на основі аналізу технічних вимог з метою досягнення оптимального балансу між обчислювальною потужністю та простотою інтеграції.

2.2.1 Вибір центрального обчислювального модуля

В якості обчислювального ядра системи застосовано одноплатний комп’ютер Raspberry Pi 3 Model B.

Чотирьохядерний процесор ARM Cortex-A53 (1.4 ГГц) та 1 ГБ оперативної пам’яті забезпечують достатню продуктивність для вирішення основного завдання – обробки відеопотоку в реальному часі та виконання логічних висновків (інференсу) за допомогою оптимізованих нейромережевих моделей у середовищі TensorFlow Lite [2].

На відміну від мікроконтролерів (наприклад, родини Arduino), які непридатні для задач комп’ютерного зору, Raspberry Pi працює під керуванням повноцінної ОС Linux (Raspberry Pi OS), що надає доступ до широкого спектру програмних бібліотек.

Зовнішній вигляд плати наведено на рисунку 2.7.

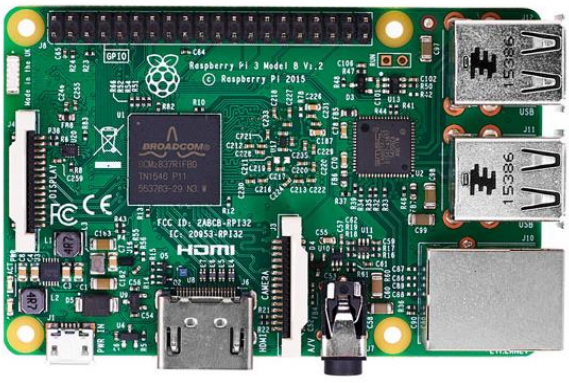
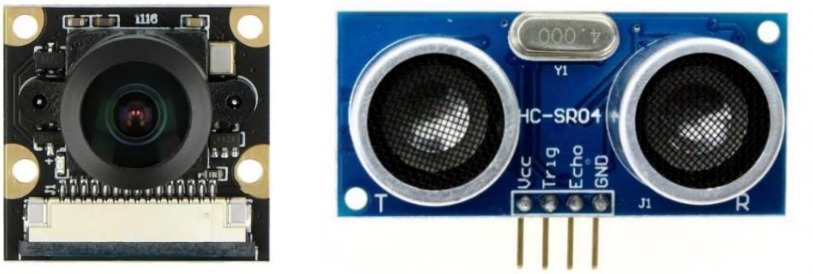


Рисунок 2.7 – Одноплатний комп’ютер Raspberry Pi 3 Model B

2.2.2 Вибір модулів сприйняття середовища

Для забезпечення надійності та всебічного аналізу оточення в системі реалізовано гібридний підхід, що поєднує два типи сенсорів: оптичний та ультразвуковий. Такий підхід гарантує отримання даних за різних умов та компенсацію недоліків кожного з методів. Зовнішній вигляд сенсорів показано на рисунку 2.8.



а) б)

Рисунок 2.8 – Модулі сприйняття середовища: а) – камера FPV Waveshare RPi Camera (G); б) – ультразвуковий давач відстані HC-SR04

Камера FPV Waveshare RPi Camera (G) використовується для отримання візуальної інформації. Ключового перевагою даної моделі є ширококутний об’єктив типу «риб’яче око» з кутом огляду 160°, що гарантує максимальне охоплення сцени перед користувачем і дозволяє виявляти бічні перешкоди. Підключення камери через апаратний інтерфейс CSI забезпечує високошвидкісну передачу даних безпосередньо до графічного процесора, мінімізуючи навантаження на CPU.

Ультразвуковий давач відстані HC-SR04 застосовується для точного безконтактного вимірювання дистанції до перешкод за принципом ехолокації [3]. Він є необхідним доповненням до камери, оскільки стабільно працює в умовах повної темряви та здатний виявляти прозорі об’єкти (наприклад, скляні двері), які є складною ціллю для алгоритмів комп’ютерного зору.

2.2.3 Вибір модулів зворотного зв’язку

Підсистема зворотного зв’язку спроектована як трьохрівнева система сповіщень для надання користувачеві повної та своєчасної інформації про оточення. Такий підхід гарантує, що найважливіші сигнали про небезпеку будуть помічені користувачем, не перевантажуючи його зайвою інформацією.

За допомогою динаміків система надає детальну інформацію про розпізнані об’єкти та відстань до них у спокійних умовах (наприклад, «Попереду людина, 3 метри»). Користувач заздалегідь формує уявлення про простір навколо нього та приймає обґрунтовані рішення щодо маршруту.

При наближенні до перешкоди на попереджувальну відстань (до 2 метрів) вмикається зумер (Buzzer), подаючи короткий звуковий сигнал. Цей невербальний сигнал слугує інтуїтивним попередженням про необхідність підвищити увагу, не перебиваючи основні голосові команди.

При критичному наближенні з перешкодою (менше 75 см) одночасно активується вібромотор та зумер, створюючи потужний тактильний та звуковий сигнал тривоги для негайної зупинки.

Вигляд модулів представлено на рисунку 2.9.

а) б) в)

Рисунок 2.9 – Модулі зворотного зв’язку: а) – динаміки; б) – зумер; в) – вібромотор

2.2.4 Вибір системи автономного живлення

Для забезпечення портативності та довготривалої автономної роботи системи застосовано готове комерційне рішення – портативний зарядний

пристрій (Power Bank). Такий підхід надає високий рівень безпеки завдяки вбудованим контролерам захисту від перезаряду та короткого замикання [13], а також простоту та надійність експлуатації.

Обраний Power Bank відповідає двом ключовим технічним вимогам: ємність не менше 10000 мАг для забезпечення роботи протягом 8 годин та вихідний струм не менше 2А при напрузі 5В, що є необхідним для стабільного живлення Raspberry Pi 3 Model B під навантаженням.

2.3 Обґрунтування вибору програмного забезпечення системи

Програмне забезпечення було обрано з урахуванням апаратної платформи Raspberry Pi 3 Model B та специфічних завдань, таких як обробка відео в реальному часі запуск нейромережевих моделей та керування периферійними пристроями. Програмний комплекс складається з операційної системи, мови програмування та набору спеціалізованих бібліотек.

2.3.1 Вибір операційної системи

В якості операційної системи (ОС) для розробленого пристрою застосовано Raspberry Pi OS (раніше відома як Raspbian).

Raspberry Pi OS розробляється та підтримується Raspberry Pi Foundation, що гарантує максимальну сумісність та оптимізацію роботи ОС під апаратні особливості мікрокомп'ютера. Це забезпечує стабільність роботи та доступ до всіх апаратних інтерфейсів "з коробки".

Система побудована на основі одного з дистрибутивів Linux – Debian. Це надає доступ до величезного репозиторію готових програмних пакетів, які легко встановлюються за допомогою менеджера пакетів apt, що значно спрощує налаштування робочого середовища.

ОС містить всі необхідні драйвери та модулі ядра для роботи з низькорівневими інтерфейсами Raspberry Pi, зокрема з GPIO [10] (для керування сенсорами та виконавчими механізмами) та CSI (для камери).

Зовнішній вигляд графічного середовища Raspberry Pi OS наведено на рисунку 2.10.

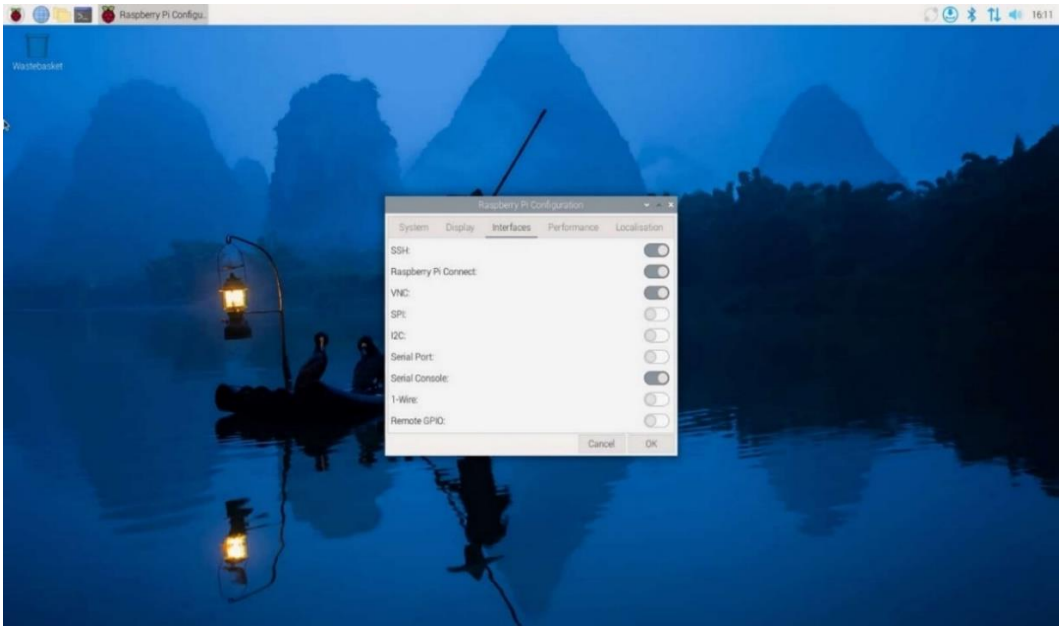


Рисунок 2.10 – Графічний інтерфейс середовища Raspberry Pi OS

2.3.2 Вибір мови програмування

Для розробки програмного забезпечення системи було обрано мову програмування Python 3. Цей вибір обґрунтований унікальним поєднанням високої швидкості розробки та доступом до потужної екосистеми спеціалізованих бібліотек [24], що є критично важливим для проектів у галузі вбудованих систем та штучного інтелекту.

Вирішальним фактором стала наявність готових, високопродуктивних та добре документованих бібліотек для всіх ключових завдань проекту. Python виступає як високорівнева мова, що дозволяє швидко та ефективно поєднувати складні модулі:

– OpenCV – для операцій з комп’ютерним зором [5];

– TensorFlow Lite – для запуску нейромережевих моделей [2];

– RPi.GPIO – для низькорівневого керування апаратними інтерфейсами [10];

– Pyttsx3 – для офлайнового синтезу мовлення [8].

Недолік Python як інтерпретованої мови, пов'язаний з нижчою швидкістю виконання, у даному проекті повністю нівелюється. Критичні за продуктивністю операції, такі як обробка зображень та математичні обчислення нейромережі, виконуються не інтерпретатором Python, а безпосередньо кодом самих бібліотек, які написані на компільованих мовах (C/C++) та оптимізовані для максимальної швидкодії.

Python 3 виступає не просто як мова програмування, а як комплексна платформа для швидкого прототипування, що дозволяє ефективно поєднати апаратне забезпечення з передовими алгоритмами комп'ютерного зору, що робить його безальтернативним вибором для даної задачі.

2.3.3 Вибір ключових програмних бібліотек

Для реалізації функціоналу системи було залучено набір спеціалізованих бібліотек, що візуально представлено на рисунку 2.11.

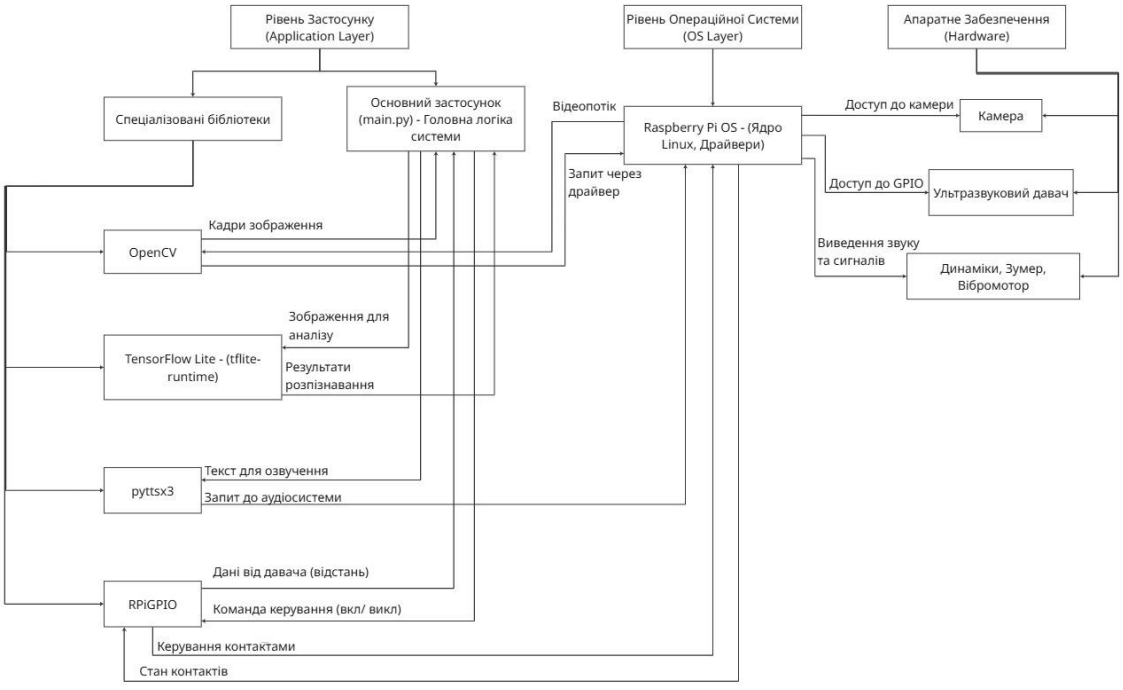


Рисунок 2.11 – Архітектура програмного забезпечення

OpenCV (Open Source Computer Vision Library) – це галузевий стандарт для завдань комп'ютерного зору. У проекті бібліотека використовується для

захоплення відеопотоку з камери (cv2.VideoCapture), зміни розміру кадрів (cv2.resize) та перетворення колірних просторів (cv2.cvtColor) на етапі передобробки зображення перед подачею в нейромережу.

TensorFlow Lite (tflite-runtime) – це оптимізований фреймворк для запуску нейромережевих моделей на вбудованих та мобільних пристроях [2]. Використання повноцінного tflite-runtime забезпечує швидкий інференс попередньо навченої моделі MobileNet SSD, що є ядром функції розпізнавання об'єктів.

RPi.GPIO – стандартна бібліотека для мови Python, що надає низькорівневий доступ до керування контактами GPIO на платі Raspberry Pi [10]. Вона використовується для відправки тригерних імпульсів на ультразвуковий давач, зчитування його відповіді, а також для подачі сигналів на зумер та транзисторний ключ вібромотора.

pyttsx3 – бібліотека для синтезу мовлення (Text-to-Speech) [8]. Її ключова перевага — повністю офлайнова робота, що є критичною вимогою для портативного асистивного пристрою [1], який повинен функціонувати незалежно від наявності інтернет-з'єднання. Вона використовує системні рушії мовлення, такі як eSpeak на Linux.

2.3.4 Створення блок-схеми алгоритму роботи системи голосового інформування

Для реалізації логіки роботи системи було розроблено комплексний алгоритм, який визначає послідовність операцій, обробки даних та прийняття рішень. В основі алгоритму лежить пріоритетний принцип, який гарантує, що реакція на загрози з найвищим рівнем небезпеки завжди виконується першочергово, забезпечуючи максимальну швидкість та безпеку для користувача.

Ключова ідея полягає в ієрархічній системі сповіщень. Замість того, щоб просто повідомляти про всі події одночасно, алгоритм аналізує ситуацію та обирає відповідний рівень реакції. Найвищий пріоритет має виявлення

перешкод на критичній відстані. У такому випадку система миттєво активує всі сигнали тривоги (звуковий та тактильний) і тимчасово ігнорує менш важливі завдання, як-от розпізнавання віддалених об'єктів. Цей підхід імітує людські рефлекси, де раптова небезпека викликає негайну реакцію, відсуваючи на другий план інші думки.

Основний принцип роботи — це безперервний цикл, в якому система, використовуючи багатопотокову архітектуру, одночасно збирає дані з усіх сенсорів (камери та ультразвукового давача). Дані надходять у чергу подій, яку головний цикл постійно обробляє. Алгоритм поєднує інформацію про розпізнаний об'єкт ("що це?") та відстань до нього ("як далеко?"), щоб сформувати єдине, змістовне повідомлення. Цей процес є повністю автономним і не вимагає втручання ззовні, що є фундаментальною вимогою до асистивного пристрою.

Алгоритм реалізовано як скінченний автомат, що постійно аналізує поточний стан середовища (наявність об'єктів, відстань) і приймає рішення про необхідну дію. Такий підхід забезпечує чіткий поділ логіки на два основні сценарії поведінки: режим інформаційного асистента у безпечних умовах та режим екстреного сповіщення при виникненні прямої загрози. Ця двоїста природа є ключовою для створення збалансованого та ефективного пристрою.

Така структурована логіка мінімізує ймовірність помилкових спрацювань та гарантує стабільну і передбачувану реакцію системи на зовнішні подразники.

Для наочної візуалізації цієї багатоетапної логіки прийняття рішень було створено блок-схему (рис. 2.12), яка точно відповідає послідовності операцій, реалізованій у програмному коді.

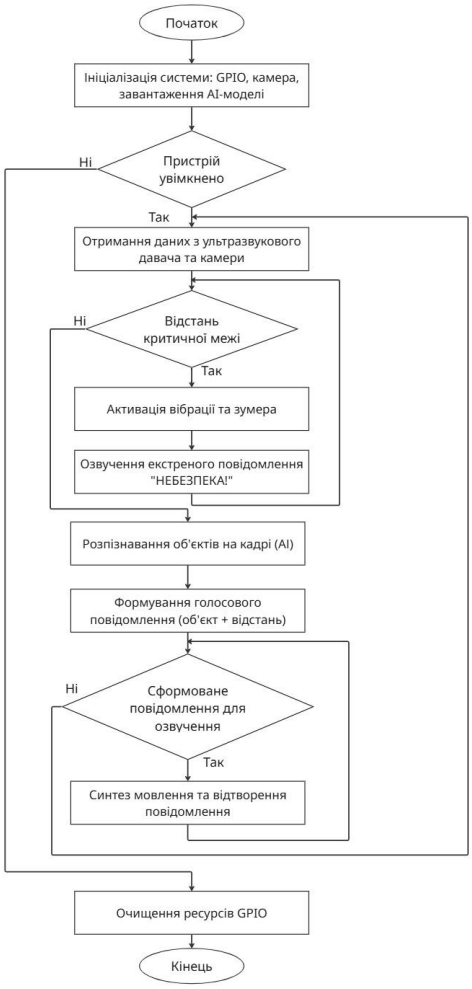


Рисунок 2.12 – Блок-схема алгоритму роботи

При запуску програми відбувається одноразова підготовка всіх компонентів до роботи. Це включає налаштування режимів роботи GPIO-пінів, завантаження в пам'ять попередньо навченої нейромережевої моделі (detect.tflite), а також створення та запуск паралельних потоків для безперервного опитування камери та ультразвукового давача.

Після ініціалізації система входить у нескінченний цикл. На кожній ітерації вона отримує з черги подій найсвіжіші дані: точну відстань до перешкоди та назву останнього розпізнаного об'єкта.

Першочергово система перевіряє, чи не є відстань до перешкоди меншою за критичну межу (наприклад, 75 см). Якщо так, система миттєво активує всі засоби тривоги — вібрацію та зумер, а також озвучує коротке екстрене повідомлення. Після цього вся інша логіка (формування

інформаційних повідомлень) ігнорується, і цикл негайно починається знову. Це гарантує максимальну швидкість реакції на пряму загрозу.

Якщо критичної небезпеки немає, система переходить до формування змістовного повідомлення. Вона аналізує, чи був розпізнаний якийсь об'єкт, і чи знаходиться він у зоні попередження (наприклад, до 2 метрів). На основі цих даних створюється текстова фраза, наприклад: "Попереду людина, на відстані близько 2 метрів" або просто "Попереду перешкода".

Якщо на попередньому етапі було сформовано непусте повідомлення, система передає його модулю синтезу мовлення (TTS), який перетворює текст на голос і відтворює його користувачеві. Якщо ніяких значущих подій не виявлено, система мовчить, щоб не відволікати користувача.

Основний цикл триває до вимкнення живлення або зупинки програми користувачем (наприклад, через Ctrl+C). Після цього виконується блок очищення, який коректно звільняє всі зайняті системні ресурси, зокрема GPIO-порти.

2.3.5 Аналіз продуктивності та шляхи оптимізації

Головним показником продуктивності є латентність — загальний час від моменту захоплення кадру до генерації голосового сповіщення. Аналіз показує, що основним обчислювальним «вузьким місцем» системи є процес інференсу (логічного висновку) нейромережевої моделі, який виконується на центральному процесорі (CPU).

З метою початкової оптимізації та забезпечення базової функціональності, у проекті реалізовано наступні стратегії:

– використання оптимізованого фреймворку [2]: замість повноцінної бібліотеки TensorFlow застосовано її полегшену версію TensorFlow Lite (tflite-runtime). Цей фреймворк розроблений для мобільних та вбудованих систем, має значно менший розмір та забезпечує прискорене виконання моделей на ARM-процесорах;

– вибір легкої архітектури моделі: в якості архітектури для

розпізнавання об'єктів було обрано MobileNet SSD. Дана архітектура спроектована для досягнення високої швидкості роботи на пристроях з низькою обчислювальною потужністю, жертвуючи незначною часткою точності на користь суттєвого підвищення швидкодії порівняно з більш важкими моделями (наприклад, YOLO або Faster R-CNN);

– застосування багатопотокової архітектури: програмна логіка, що використовує паралельні потоки для опитування давачів, не прискорює сам процес інференсу, але кардинально підвищує чутливість системи. Це дозволяє миттєво реагувати на критичні перешкоди, виявлені ультразвуковим давачем, не чекаючи завершення довготривалого процесу розпізнавання об'єкта.

Для подальшого підвищення продуктивності та зменшення латентності системи можлива реалізація наступних шляхів оптимізації:

– апаратне прискорення: найбільш ефективний метод, що передбачає використання зовнішнього співпроцесора для нейронних обчислень, наприклад, Google Coral USB Accelerator. Такі пристрої містять спеціалізований TPU (Tensor Processing Unit), який здатний виконувати операції інференсу в десятки разів швидше, ніж CPU Raspberry Pi.

– подальша оптимізація моделі: застосування техніки повної цілочисельної квантизації (Full Integer Quantization). Цей процес перетворює вагові коефіцієнти моделі з 32-бітних чисел з плаваючою комою (FP32) на 8-бітні цілі числа (INT8). Це призводить до зменшення розміру моделі приблизно в 4 рази та прискорення її виконання на CPU в 2-3 рази при мінімальній втраті точності;

– оптимізація конвеєра обробки даних: аналіз та оптимізація етапу передобробки зображення (зміна розміру, нормалізація) та використання більш ефективних бібліотек для завантаження й аргументації даних.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Програмна реалізація системи

Програмне забезпечення для комп’ютеризованої системи голосового інформування розроблено мовою програмування Python 3. В основі архітектури лежить багатопотоковий підхід (multi-threading) з використанням черги подій (queue.Queue) для обміну даними між потоками. Така архітектура обрана для забезпечення максимальної чутливості системи до раптових перешкод: швидкий процес вимірювання відстані виконується паралельно з більш повільним та ресурсоємним процесом розпізнавання об'єктів, не блокуючи один одного.

На початку програмного коду визначено набір глобальних констант, що дозволяє гнучко налаштовувати систему без втручання в основну логіку. Це включає призначення GPIO-пінів, пороги відстані, шляхи до файлів моделі [2] та поріг впевненості для розпізнавань, як показано на рисунку 3.1.

```
# Налаштування GPIO
GPIO_TRIGGER = 20
GPIO_ECHO = 21
GPIO_VIBRATOR = 18
GPIO_BUZZER = 17

# Константи відстаней
DISTANCE_CRITICAL_CM = 75
DISTANCE_WARN_CM = 200
MAX_DETECTION_DISTANCE_CM = 400

# Налаштування моделі
MODEL_PATH = 'detect.tflite'
LABEL_PATH = 'labelmap.txt'
MIN_CONFIDENCE = 0.30
```

Рисунок 3.1 – Лістинг конфігураційних констант та налаштування пінів

Для генерації голосових повідомлень реалізовано клас VoiceAssistant, що інкапсулює роботу з бібліотекою pyttsx3 [8]. При ініціалізації він автоматично намагається активувати системний український голосовий пакет. Важливою особливістю класу є механізм захисту від повторів, який не дозволяє озвучувати одне й те саме повідомлення частіше, ніж раз на 5 секунд, що робить взаємодію з пристроєм більш комфортною. Для уникнення конфліктів при одночасних запитах на озвучення використовується об'єкт блокування threading.Lock. Реалізацію цього механізму представлено на рисунку 3.2.

Рисунок 3.2 – Лістинг реалізації методу озвучення з захистом від повторів

Збір даних про оточення виконується асинхронно у двох незалежних фонових потоках. Перший потік, ultrasonic_worker [3] (рис. 3.3), працює у швидкому циклі, вимірює відстань за допомогою давача HC-SR04 і надсилає результат у спільну чергу подій. Другий потік, camera_worker, виконує складнішу задачу: безперервно захоплює кадри з камери, обробляє їх та проводить розпізнавання об'єктів за допомогою моделі TensorFlow Lite [2], після, надсилає результат у чергу. Така архітектура повністю роз'єднує (decouple) процеси збору даних, ресурсоємна задача розпізнавання об'єктів не блокує миттєву передачу критично важливих даних про відстань до перешкоди.

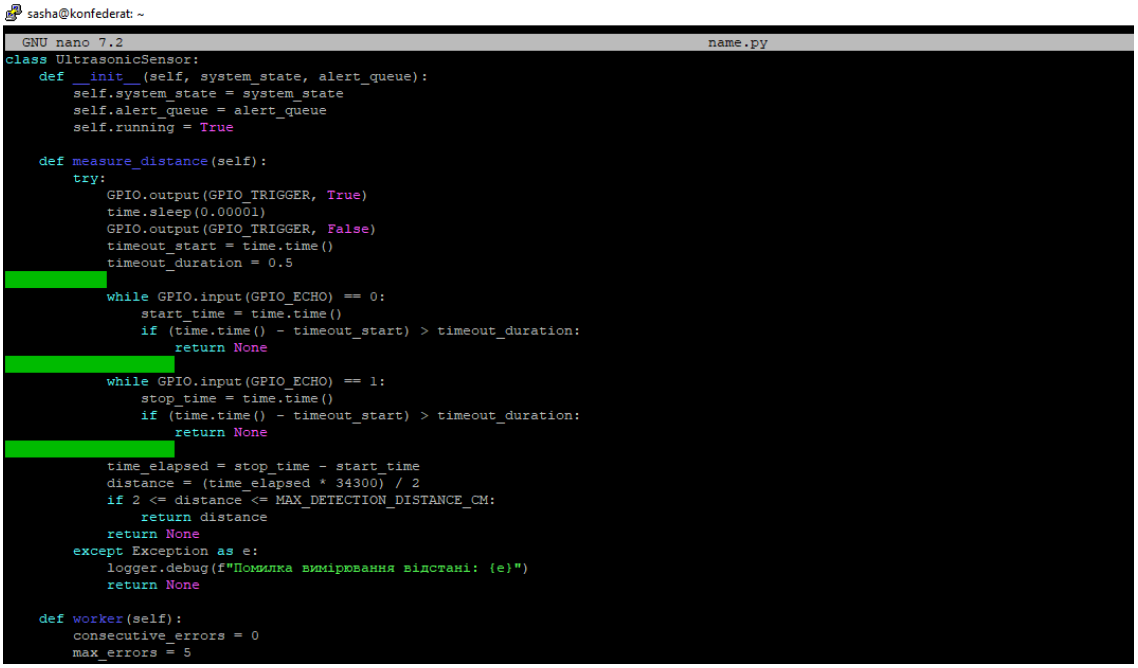


Рисунок 3.3 – Лістинг реалізації фонового потоку для ультразвукового давача

Центральним елементом програмної логіки є головний цикл у функції main(), реалізований за подієво-орієнтованим принципом. Його робота базується на безперервній обробці подій, що надходять зі спільної черги (Queue). Виклик alert_queue.get() є блокуючою операцією, що дозволяє циклу ефективно «спати», не споживаючи ресурси процесора, доки один з фонових потоків не надішле нові дані. Отримавши подію, програма оновлює відповідну змінну стану — current_distance або current_object, підтримуючи таким чином актуальну картину навколишнього середовища.

Після оновлення стану виконується алгоритм прийняття рішень, побудований на основі чіткої ієрархії пріоритетів. Найвищий пріоритет надається перевірці на критичну небезпеку (рис. 3.4): якщо поточна відстань менша за встановлений поріг (DISTANCE_CRITICAL_CM), система негайно активує всі засоби тривоги (вібрацію, зумер та екстрене голосове повідомлення). Після цього, за допомогою логіки, аналогічної оператору continue, виконання поточної ітерації циклу припиняється, і система негайно переходить до очікування наступної події. Якщо ж критичної загрози немає, алгоритм переходить до формування змістовного інформаційного

повідомлення, поєднуючи дані про розпізнаний об'єкт та відстань до нього, яке потім передається на озвучення.

```
# Початкове повідомлення
voice.speak("Система навігації запущена", force=True)

last_message_time = 0

try:
    while True:
        try:
            alert = alert_queue.get(timeout=0.5)
        except:
            continue

        current_time = time.time()

        # Обробка помилок системи
        if alert['type'] == 'error':
            voice.speak(f"Системна помилка: {alert['value']}", force=True)
            logger.error(alert['value'])
            continue

        # Критична відстань - негайне попередження
        if system_state.ultrasonic_active and system_state.distance < DISTANCE_CRITICAL_CM:
            alert_system.activate_critical_alert()
            voice.speak("Увага! Критична відстань!", force=True)
            time.sleep(0.5)
            alert_system.deactivate_alert()
            continue
        else:
            alert_system.deactivate_alert()

        # Регулярні голосові повідомлення
        if current_time - last_message_time >= 2:
            message = generate_voice_message(system_state)
            if message:
                voice.speak(message)
                last_message_time = current_time
```

Рисунок 3.4 – Лістинг фрагменту коду з реалізацією пріоритетної обробки

Для забезпечення стабільності та коректного завершення роботи вся логіка програми обгорнута в блок try...finally (рис. 3.5). Це гарантує, що при зупинці програми (наприклад, через Ctrl+C) буде виконано команду GPIO.cleanup() [10], яка звільняє зайняті GPIO-порти та запобігає помилкам при наступних запусках скриптів.

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

Рисунок 3.5 – Лістинг структури запуску головного циклу та коректне завершення

Нижче представлена блок-схема, що ілюструє загальний алгоритм роботи програмного забезпечення (рис. 3.6).

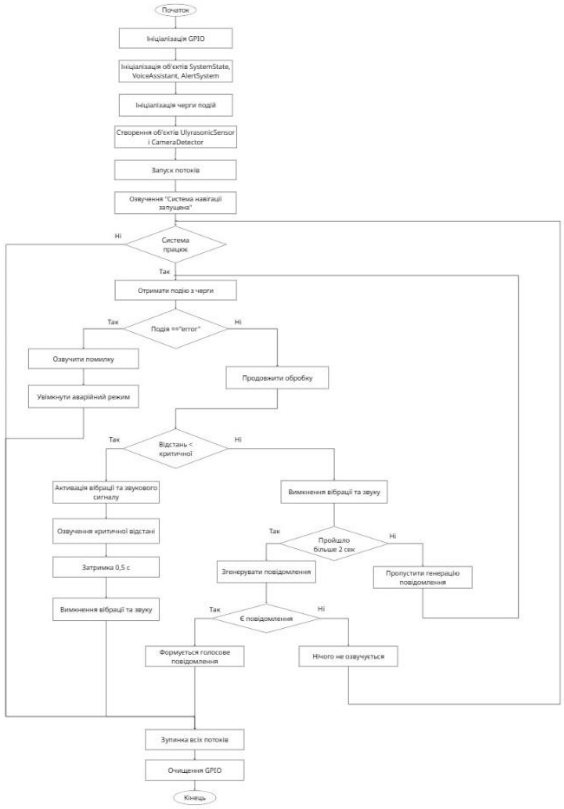


Рисунок 3.6 – Блок-схема алгоритму програмного забезпечення

3.2 Апаратна реалізація системи

Апаратна реалізація системи була зосереджена на створенні діючого макету (прототипу). Головною метою цього етапу була практична перевірка розробленої електричної схеми, тестування взаємодії всіх апаратних модулів між собою та з розробленим програмним забезпеченням. Для цього було використано безпайкову макетну плату (breadboard), що дозволило гнучко та швидко збирати схему, вносити зміни та проводити діагностику без необхідності паяння, що є оптимальним для етапу розробки.

Основою макету слугував мікрокомп'ютер Raspberry Pi, контакти живлення якого (5V та GND) були підключені до відповідних силових ліній на макетній платі. Це дозволило централізовано подавати живлення на всі периферійні модулі. Керування та обмін даними з модулями здійснювався через GPIO-порти, з'єднані з компонентами за допомогою спеціальних дротів Dupont.

Ультразвуковий давач HC-SR04 був встановлений на макетній платі. Його вивід VCC було підключено до лінії живлення 5В, а GND — до лінії землі. Сигнальні виводи Trig та Echo були з'єднані з відповідними GPIO-пінами Raspberry Pi (GPIO 23 та GPIO 24). Для захисту вхідного порту мікрокомп'ютера від 5-вольтового сигналу з піна Echo, безпосередньо на макетній платі була зібрана схема дільника напруги [9]: резистор номіналом 1 кОм було встановлено послідовно на лінію Echo, а резистор 2 кОм — між цією лінією та землею, що забезпечило зниження напруги до безпечного рівня 3.3В.

Для реалізації трьохрівневої системи сповіщень на макетну плату було встановлено зумер та схему керування вібромотором. Активний зумер було підключено безпосередньо: його позитивний вивід — до GPIO 27, а негативний — до лінії землі. Для керування вібромотором, який є індуктивним навантаженням, було зібрано схему транзисторного ключа. NPN-транзистор BC547 було встановлено на платі; його база була підключена через обмежувальний резистор 1 кОм до GPIO 17, емітер — до землі, а колектор — до негативного виводу вібромотора. Позитивний вивід мотора було підключено до лінії живлення 3.3В.

Для виведення голосових повідомлень плата зовнішнього аудіопідсилювача отримувала живлення від силових ліній 5В та GND на макетній платі, а її аудіовхід було з'єднано з 3.5 мм роз'ємом Raspberry Pi за допомогою AUX-кабелю. Динаміки, у свою чергу, підключались до вихідних клем підсилювача. Загальне живлення всього прототипу забезпечувалось шляхом підключення Power Bank до Micro-USB порту Raspberry Pi.

Результатом даного етапу є повністю функціональний лабораторний стенд, який дозволяє продемонструвати роботу системи в реальному часі. Зовнішній вигляд зібраного прототипу на макетній платі показано на рисунку 3.7.

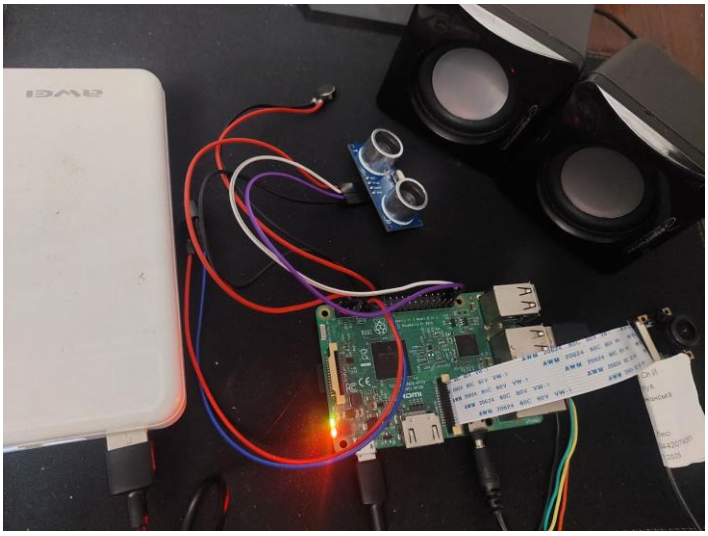


Рисунок 3.7 – Апаратна реалізація системи голосового інформування

3.3 Тестування системи голосового інформування

Після завершення апаратної реалізації прототипу на макетній платі було проведено етап налаштування програмного забезпечення та комплексного тестування системи. Метою тестування була перевірка працездатності всіх функціональних вузлів, коректності роботи розроблених програмних алгоритмів та відповідності фінального прототипу поставленим технічним вимогам.

Підготовчий етап включав розгортання програмного коду на апаратному стенді. Всі розроблені Python-скрипти та необхідні файли, такі як модель detect.tflite, мітки labelmap.txt та конфігураційний файл, були завантажені на MicroSD карту Raspberry Pi у відповідну робочу директорію (рис. 3.8.).

```
sasha@konfederat:~ $ mkdir -p ~/navigation_system/logs
mv ~/detect.tflite ~/navigation_system/
mv ~/labelmap.txt ~/navigation_system/
mv ~/name.py ~/navigation_system/
mv ~/name1.py ~/navigation_system/
mv ~/names.py ~/navigation_system/
mv ~/test.py ~/navigation_system/
touch ~/navigation_system/logs/system.log
sasha@konfederat:~ $ ls -l ~/navigation_system
ls -l ~/navigation_system/logs
total 4136
-rw-r----- 1 sasha sasha 4183312 Jun 29  2018 detect.tflite
-rw-r----- 1 sasha sasha    2715 Jun 23 07:01 labelmap.txt
```

Рисунок 3.8. – Завантажені Python-скрипти

Перед запуском було проведено фінальне налаштування параметрів у конфігураційному файлі, де особливу увагу було приділено експериментальному підбору порогів відстані (DISTANCE_CRITICAL_CM та DISTANCE_WARN_CM) та порогу впевненості для нейромережі (MIN_CONFIDENCE). Значення 0.60 було обрано як оптимальний баланс між чутливістю розпізнавання та мінімізацією хибних спрацювань. Перший запуск основного скрипта підтвердив відсутність помилок ініціалізації та коректну роботу модуля синтезу мовлення, що повідомив про готовність голосовим сигналом «Система навігації запущена» (рис. 3.9).

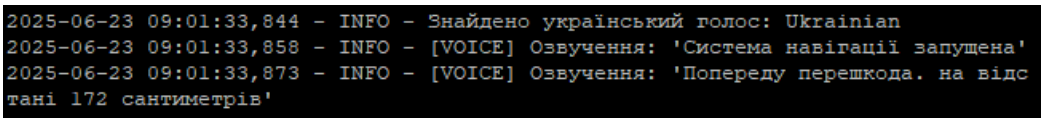


Рисунок 3.9 – Початок запуску роботи системи

Після успішного налаштування було проведено комплексне тестування за кількома сценаріями для перевірки кожного функціонального блоку окремо та їхньої спільної роботи. Спочатку тестувався модуль вимірювання відстані та сповіщень про небезпеку (рис. 3.10). При наближенні об'єкта до ультразвукового давача система стабільно генерувала голосове повідомлення "Попереду перешкода" при перетині межі у 200 см, а при критичному зближенні (менше 75 см) коректно активувала пріоритетний сигнал тривоги, що включав вібрацію, звуковий сигнал зумера та екстрене голосове повідомлення, перериваючи будь-які інші інформаційні повідомлення.

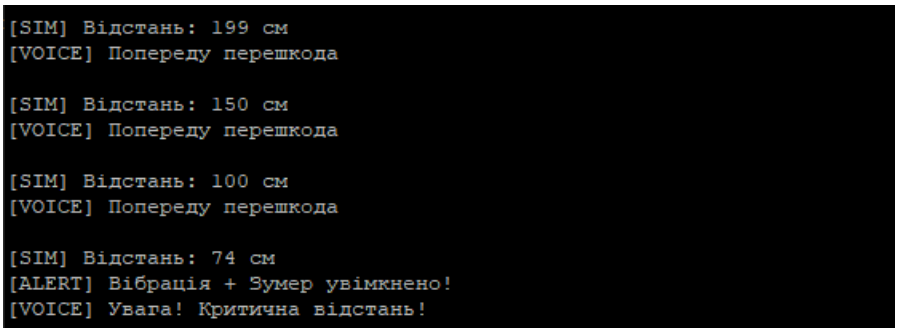


Рисунок 3.10 – Тестування вимірювання відстані та сповіщень про небезпеку

Наступним етапом було тестування модуля розпізнавання об'єктів (рис. 3.11.). Перед камерою на відстані 1-3 метри розміщувалися різні предмети, класи яких присутні у наборі даних COCO. Система успішно та стабільно ідентифікувала такі об'єкти, як: людина (person), ноутбук (laptop), чашка (cup), книга (book) та мобільний телефон (cell phone) [6]. Розпізнавання відбувалося з рівнем впевненості, що перевищував встановлений поріг у 60%, а об'єкти, що не належать до відомих класів, коректно ігнорувалися.

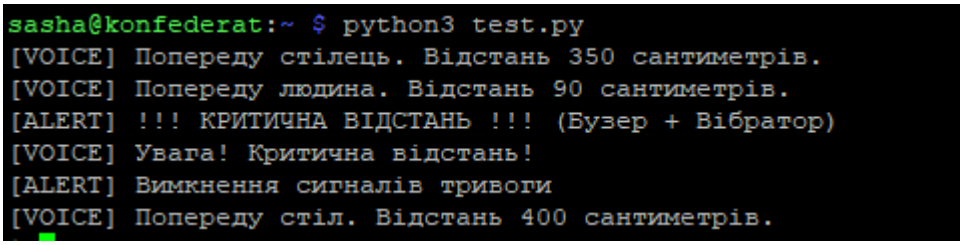


Рисунок 3.11 – Тестування розпізнавання об'єктів

Проведене тестування в сукупності підтвердило повну працездатність розробленого прототипу. Апаратна частина функціонує стабільно, а програмне забезпечення коректно реалізує закладені алгоритми, включаючи багатопотокову обробку даних та пріоритезацію сповіщень. Таким чином, система успішно вирішує поставлене завдання з інформування користувача про об'єкти та перешкоди в його оточенні.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Характеристика життєдіяльності людини у системі «людина – машина – середовище існування»

Система «людина – машина – середовище існування» (Л-М-С) є фундаментальною концепцією в галузі безпеки життєдіяльності та ергономіки [16]. Вона розглядає будь-яку трудову або побутову діяльність не як набір окремих елементів, а як єдиний комплекс, де всі складові тісно взаємопов’язані. Головна мета аналізу цієї системи — оптимізувати взаємодію її компонентів для досягнення максимальної ефективності, комфорту та безпеки для людини.

Система складається з трьох основних елементів [16]:

1) людина є центральним і найважливішим елементом системи, оскільки ставить мету, приймає рішення та керує процесом. Вона виступає як керуюча ланка, що отримує інформацію, аналізує її та впливає на "машину". У контексті КС голосового інформування, «людиною» є як розробник на етапі створення, так і кінцевий користувач з вадами зору на етапі експлуатації. Характеристики людини, що впливають на роботу системи, включають:

– психофізіологічні властивості: швидкість сенсомоторних реакцій на зовнішні показники (голосові та тактильні сигнали пристрою), гострота слуху та дотику (важливо для кінцевого користувача), координація рухів, витривалість, рівень втоми;

– психологічні властивості: рівень уваги, пам'ять, мотивація, емоційний стан, стресостійкість, рівень професійної підготовки та навичок;

– антропометричні дані: зріст, розміри тіла, довжина рук, фізична сила, що визначає зручність доступу до органів керування "машиною".

2) під "машиною" розуміють будь-який технічний об'єкт, з яким взаємодіє людина для досягнення своєї мети. У даному випадку це розроблений пристрій для голосового інформування. Основні характеристики "машини":

– технічні параметри: продуктивність, точність давачів, надійність програмного забезпечення, потужність;

– ергономічні параметри: зручність та логічність розташування органів керування (тактильно розрізнювані кнопки), інформативність голосових повідомлень, відповідність конструкції (вага, розмір) антропометричним даним людини;

– параметри безпеки: наявність захисних механізмів (корпус пристрою, захисні покриття на платах та правильне кріплення елементів), систем аварійного вимкнення, рівень шуму, вібрації та електромагнітного випромінювання [23].

3) сукупність зовнішніх умов, в яких функціонує система «людина – машина», охоплює фізичні, соціальні та інформаційні фактори, що впливають на взаємодію людини з технічним засобом. Середовище може як сприяти, так і перешкоджати ефективній та безпечній діяльності. Його поділяють на:

– виробниче середовище: умови в лабораторії або цеху, де створюється чи обслуговується пристрій;

– середовище експлуатації: умови, в яких кінцевий користувач використовує пристрій. Для системи голосового інформування це динамічне міське середовище, приміщення, громадський транспорт.

Ефективність та безпека життєдіяльності в системі Л-М-С залежать від гармонійної взаємодії її елементів. Недосконаліть "машини" (нечітке голосове повідомлення пристрою) може призвести до помилки "людини" (неправильної реакції користувача) у складному "середовищі" (на перехресті вулиць).

Комплексний аналіз системи «людина – машина – середовище існування» дозволяє виявити потенційні небезпеки на всіх етапах життєвого

циклу технічного засобу та розробити заходи для їх усунення або мінімізації, забезпечуючи безпечні та комфортні умови для людини.

4.2 Вплив електромагнітних полів (ЕМП) на людину та заходи щодо зменшення їх впливу

Людина постійно перебуває під дією електромагнітних полів (ЕМП), які створюються як природними джерелами (магнітне поле Землі, сонячне випромінювання), так і численними штучними [14]. До штучних джерел належать лінії електропередач, промислове обладнання, побутові прилади, теле- та радіостанції, а також системи бездротового зв'язку, такі як мобільні телефони, Wi-Fi та Bluetooth. Рівень впливу ЕМП на організм залежить від їхньої інтенсивності (потужності), частоти, тривалості дії та індивідуальних особливостей організму.

Електромагнітні поля, з якими стикається людина в процесі життєдіяльності, умовно поділяють на кілька діапазонів. У контексті роботи з розробленою системою голосового інформування значення мають:

1) поля промислової частоти (50 Гц): створюються блоками живлення комп'ютерів під час розробки [17];

2) поля радіочастотного діапазону (RF): генеруються пристроями бездротового зв'язку. До них належать модулі Wi-Fi та Bluetooth, які є частиною розробленої системи, а також мобільні телефони та роутери.

Важливо розрізняти неіонізуюче та іонізуюче випромінювання [4]. ЕМП від побутових та комп'ютерних пристроїв належать до неіонізуючого типу, тобто їхня енергія недостатня для того, щоб викликати іонізацію атомів у клітинах організму, на відміну від рентгенівського чи гамма-випромінювання.

Вплив ЕМП на біологічні об'єкти може мати тепловий та нетепловий характер:

1) тепловий ефект виникає при дії ЕМП високої інтенсивності, що призводить до нагрівання тканин організму. Санітарні норми ДСанПіН

України розроблені для обмеження теплового ефекту до безпечного рівня [18]. Для малопотужних пристроїв, включаючи розроблену в роботі систему голосового інформування, тепловий ефект є незначним і не становить загрози;

2) нетепловий (біологічний) ефект. Виникає при тривалій дії ЕМП низької інтенсивності. Наукові дослідження вказують на можливий вплив на найбільш чутливі системи організму[14] :

– нервова система: підвищена втомлюваність, дратівливість, порушення сну, головний біль;

– серцево-судинна та імунна системи: гіпотеза про можливий вплив на ритм серцевих скорочень та ослаблення імунної відповіді організму.

Для захисту від потенційного негативного впливу ЕМП застосовується комплекс заходів, заснованих на трьох основних принципах [23]:

1) захист часом. Метод полягає у зменшенні тривалості контакту з джерелом ЕМП. Для користувача розробленого пристрою цей захист реалізовано через можливість увімкнення та вимкнення системи за потреби;

2) захист відстанню. Інтенсивність ЕМП стрімко зменшується зі збільшенням відстані. Хоча розроблений пристрій носиться близько до тіла, його потужність настільки мала, що це є безпечним;

3) екранування джерел ЕМП. Метод передбачає використання матеріалів, що поглинають або відбивають випромінювання:

– технічне екранування: корпуси сучасної техніки, включаючи корпус розробленого пристрою, та правильне заземлення обладнання значно знижують рівень полів;

– використання спеціальних матеріалів: у промисловості використовуються спеціальні тканини, сітки або фарби для екранування приміщень;

4) організаційні заходи. До них належить використання сертифікованого обладнання, що відповідає чинним санітарним нормам. У проекті використовуються сертифіковані компоненти (Raspberry Pi), що гарантує їхню безпеку.

ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи було успішно досягнуто поставленої мети — розроблено та протестовано діючий прототип комп'ютеризованої системи голосового інформування для людей з вадами зору. Проект демонструє практичне застосування сучасних вбудованих систем та технологій штучного інтелекту для вирішення важливого соціального завдання з підвищення безпеки та автономності людей з порушеннями зору.

В рамках роботи було проведено комплексний аналіз існуючих асистивних технологій, на основі якого було сформульовано технічні вимоги до системи. Була спроектована модульна апаратна архітектура, ядром якої є мікрокомп'ютер Raspberry Pi 3 Model B. Обґрунтовано вибір компонентної бази, що включає ширококутну камеру Waveshare RPi Camera (G) для візуального аналізу, ультразвуковий давач HC-SR04 для вимірювання відстані та трьохрівневу систему зворотного зв'язку (динаміки, зумер, вібромотор). Програмне забезпечення було реалізовано мовою Python з використанням багатопотокової архітектури та черги подій, що забезпечило високу чутливість системи до загроз.

Практична частина роботи полягала у створенні макету пристрою та його всебічному тестуванні. Проведені експерименти підтвердили повну працездатність системи. Прототип стабільно виявляє перешкоди на різній відстані, коректно розпізнає базові об'єкти за допомогою нейромережевої моделі TensorFlow Lite та генерує відповідні, пріоритезовані сповіщення. Було підтверджено ефективність ієрархічної системи сповіщень: від інформативних голосових повідомлень до екстрених тактильних та звукових сигналів тривоги.

Таким чином, всі завдання, поставлені на початку роботи, були повністю виконані. Створений прототип є функціональним лабораторним стендом, який доводить життєздатність обраної концепції. Результати даної роботи можуть бути використані як основа для подальших наукових досліджень та розробки

комерційного асистивного пристрою.

Напрямками подальшого розвитку проєкту є вдосконалення апаратної частини шляхом створення компактного ергономічного корпусу, а також розширення програмного функціоналу: інтеграція GPS-модуля для навігації, додавання функції розпізнавання голосових команд та використання більш досконалих нейромережевих моделей для розпізнавання ширшого кола об'єктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dutoit, T. An Introduction to Text-to-Speech Synthesis. Springer Science & Business Media, 1997. 294 p.

2. Getting Started with TensorFlow Lite on Raspberry Pi [Electronic resource]. – TensorFlow Official Documentation. – Access mode: https://www.tensorflow.org/lite/guide/raspberry_pi (дата звернення 02.06.2025р)

3. HC-SR04 Ultrasonic Sensor Datasheet [Electronic resource]. – Access mode: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf (дата звернення 04.05.2025р).

4. Institute of Electrical and Electronics Engineers (IEEE). IEEE Standard for Safety Levels with Respect to Human Exposure to Radio Frequency Electromagnetic Fields, 3 kHz to 300 GHz. IEEE Std C95.1, 2019.

5. Joseph Howse, Joe Minichino. Learning OpenCV 4 Computer Vision with Python 3. – 3rd ed. – Packt Publishing, 2020. 482 p.

6. Narmada A., Bashetty V., Pentala P., Balabhakthula N. Raspberry Pi-Based Design and Implementation of Object Detection for Visually Impaired. Advanced Engineering Science, 2022. Vol. 54, Issue 02. – P. 1241-1250.

7. Palamar A., Karpinski M., Palamar M., Osukhivska H., Mytnyk M. Remote Air Pollution Monitoring System Based on Internet of Things. CEUR Workshop Proceedings, 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022), Ternopil, Ukraine, November 22–24, 2022. Vol. 3309. P. 194-204.

8. pyttsx3 2.90: Offline Text to Speech library for Python [Electronic resource]. – The Python Package Index (PyPI). – Access mode: https://pypi.org/project/pyttsx3/ (дата звернення 03.06.2025).

9. Simon Monk . Raspberry Pi Cookbook. – 3rd ed. – O'Reilly Media, 2019. – 464 p.

10. RPi.GPIO Module Documentation [Electronic resource]. – Access mode: https://pypi.org/project/RPi.GPIO/ (дата звернення 08.05.2025).

11. J. A. Smith, K. L. Jones. U.S. Patent No. 9,454,789. Portable electronic device for assisting visually impaired persons with object identification and navigation, 2016.

12. Velychko D., Osukhivska H., Palaniza Y., Lutsyk N., Sobaszek L. Artificial Intelligence Based Emergency Identification Computer System. Advances in Science and Technology Research Journal, 18 no. 2, 2024, P. 296-304.

13. Voloskyi V., Leshchyshyn Y., Romanyshyn N., Palamar A., Tarasenko L. Method and algorithm for efficient cell balancing in the lithium-ion battery control system. CEUR Workshop Proceedings, The 1st International Workshop on Bioinformatics and Applied Information Technologies (BAIT 2024), Zboriv, Ukraine, October 02-04, 2024. Vol. 3842. P. 258-267.

14. World Health Organization (WHO). Electromagnetic fields and public health: mobile phones. Fact sheet No. 193, 2014.

15. Бедрій Я. І., Джигирей В. С., Кислицина А. І. Безпека життєдіяльності: навчальний посібник. – Київ: Центр учбової літератури, 2018. 288 с.

16. Гандзюк М.П., Безпека життєдіяльності: навч. посіб. – К.: Центр учбової літератури, 2013. 256 с.

17. Державна служба України з надзвичайних ситуацій. Методичні рекомендації з оцінки умов праці при дії ЕМП.

18. ДСанПіН 3.3.6.096-2002 «Державні санітарні норми і правила захисту населення від впливу електромагнітних полів» – Міністерство охорони здоров’я України.

19. Жаровський Р.О., Луцик Н.С., Осухівська Г.М., Паламар А.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 123 «Комп’ютерна інженерія» усіх форм навчання. Тернопіль: ТНТУ, 2024. 39 с.

20. Жидецький В. Ц. Основи охорони праці: підручник. Львів: Афіша, 2005. 347 с.

21. Карунакаран, С., Раджа, К. Система голосових сповіщень на основі Raspberry Pi для ідентифікації об'єктів. Вісник інженерії та прикладних наук, 2021. Т. 8, № 3. С. 45–51.

22. Коваленко, І. І., & Петренко, В. М. Особливості синтезу українського мовлення для вбудованих асистивних систем. Матеріали V Міжнародної науково-практичної конференції «Інформаційні технології в освіті, науці і техніці». – Черкаси: ЧДТУ, 2022. С. 112–114.

23. Литвиненко О.М., Основи охорони праці та безпеки життєдіяльності: навч. посіб. – Харків: ХНАМГ, 2014. 304 с.

24. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп’ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ. 2019. 150 с.

25. Харченко О., Яцишин В. Розробка та керування вимогами до програмного забезпечення на основі моделі якості. Вісник ТДТУ. Тернопіль, 2009. Т. 14. №1. С. 201-207.

26. Ясінський Р.В., Осухівська Г.М., Паламар А.М., Величко Д.В. Комп’ютерна система для контролю параметрів мікроклімату теплиць на основі інтернету речей. Актуальні задачі сучасних технологій : збірник тез доповідей XI міжнародної науково-практичної конференції молодих учених та студентів (Тернопіль, 7-8 грудня 2022 року), Тернопіль: ФОП Паляниця В. А., 2022. С. 177.

27. Яцишин В., Пастух О., Жаровський Р. Technology of relational database management systems performance evaluation during computer systems design. Scientific Journal of TNTU, Ternopil, Ukraine, 2023. Vol. 109, No 1. P. 54–65.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

"Затверджую"

Завідувач кафедри КС

_____ Осухівська Г.М.

"___" _____2025 р

КОМП'ЮТЕРИЗОВАНА СИСТЕМА ГОЛОСОВОГО ІНФОРМУВАННЯ ЛЮДЕЙ З ВАДАМИ ЗОРУ НА БАЗІ RASPBERRY PI

ТЕХНІЧНЕ ЗАВДАННЯ

на _8_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІ-41
_____ к.т.н., ст.викл. Стадник Н.Б.	_____ Конфедерат О.О.
«____» _____ 2025 р.	«____» _____ 2025 р.

Тернопіль 2025

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютеризована система голосового інформування людей з вадами зору на базі Raspberry PI».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.212.00.00

1.2 Виконавець

Студент групи СІ-41, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Конфедерат О.О.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№4/7-53 від 27.01.2025 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 27.01.2025 р.

Плановий термін завершення виконання кваліфікаційної роботи – 23.06.2025 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ISO, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи. Попередній захист кваліфікаційної роботи відбувається при готовності роботи – наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Система, що розробляється призначена для:

– виявлення статичних та динамічних перешкод у режимі реального часу;

– голосового інформування користувача про тип та відстань до перешкод;

– підвищення безпеки та автономності пересування людей з вадами зору.

2.2 Мета створення системи

Метою кваліфікаційної роботи є розробка та тестування портативної комп'ютеризованої системи голосового інформування для людей з вадами зору на основі мікрокомп'ютера Raspberry Pi.

2.3 Характеристика об'єкту

Розроблювана система є портативним асистивним пристроєм індивідуального користування, призначеним для використання як у приміщеннях, так і на відкритій місцевості.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

3.1.1 Вимоги до структури та функціонування системи

Система повинна складатись з:

– апаратного модуля (мікрокомп’ютер, сенсори, засоби зворотного зв’язку);

– спеціалізованого програмного забезпечення для обробки даних та генерації сповіщень.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Зв'язок між апаратними компонентами (ультразвуковий давач, зумер, вібромотор) та мікрокомп'ютером здійснюється через інтерфейс GPIO. Камера підключається через спеціалізований порт CSI. Зв'язок з користувачем – через аудіовихід (голосові команди) та тактильні/звукові сигнали.

3.1.3 Вимоги до режимів функціонування системи

Для системи визначено два режими функціонування:

– нормальний режим функціонування;

– аварійний режим функціонування.

Основним режимом функціонування є нормальний режим, при якому система використовує всі сенсори (камеру та ультразвуковий давач) для комплексного аналізу оточення.

Аварійний режим функціонування характеризується відмовою одного з компонентів, наприклад, камери. При цьому система повинна продовжувати виконувати базову функцію виявлення перешкод за допомогою ультразвукового давача.

3.1.4 Вимоги по діагностуванню системи

Для діагностування системи використовуються інструменти, вмонтовані в операційну систему, а також логування роботи програмного забезпечення. Система повинна забезпечувати можливість перегляду діагностичних повідомлень та логів роботи для виявлення можливих збоїв.

3.1.5 Перспективи розвитку, проектування системи

Дана система може бути розширена завдяки додаванню наступних функцій:

– інтеграція GPS-модуля для навігації по маршруту;

– вдосконалення моделей розпізнавання об’єктів для ідентифікації більшої кількості предметів;

– додавання функції розпізнавання голосових команд для керування системою.

3.2 Показники призначення

Система повинна передбачати можливість програмного та апаратного масштабування. Можливості масштабування повинні забезпечуватися засобами використовуваного базового програмного і технічного забезпечення.

3.2.1 Вимоги до надійності

Система повинна забезпечувати працездатність та відновлення своїх функцій при виникненні наступних ситуацій:

– при збоях в системі електропостачання (тимчасове відключення);

– при помилках в роботі окремих апаратних засобів;

– при помилках, пов'язаних з програмним забезпеченням.

Для захисту апаратури від нестабільного живлення та просідань напруги від акумулятора застосовуються вбудовані в Power Bank контролери.

3.3 Вимоги до безпеки

Зовнішні елементи технічних засобів системи, що перебувають під напругою, повинні мати захист від випадкового дотику.

Система живлення повинна забезпечувати захисне вимикання при перевантаженнях і коротких замиканнях.

Загальні вимоги пожежної безпеки повинні відповідати нормам на побутове електрообладнання.

3.3.1 Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Мікроклімат в приміщеннях повинен відповідати нормам виробничого мікроклімату по ДСН 3.3.6.042-99:

– температуру повітря в межах від +10°С до +35°С;

– відносну вологість повітря при 25°С в межах від 30% до 80%;

– атмосферний тиск 760±25 мм рт. ст.

Періодичне технічне обслуговування використовуваних технічних засобів має проводитися відповідно до вимог технічної документації, але не рідше ніж один раз на рік.

Періодичне технічне обслуговування і тестування технічних засобів повинні включати обслуговування і тестування всіх використовуваних засобів, датчики, контроллери, системи передачі даних, пристрої безперебійного живлення.

На підставі результатів тестування технічних засобів повинні проводитися аналіз причин виникнення виявлених дефектів і прийматися заходи по їх ліквідації.

3.4 Вимоги до захисту інформації від несанкціонованого доступу

Система не зберігає та не обробляє персональні чи конфіденційні дані користувача, спеціальні вимоги до захисту інформації не висуваються.

Основною вимогою є захист програмного коду та файлів налаштувань від несанкціонованої зміни, що забезпечується стандартними засобами операційної системи (персональний доступ).

3.4.1 Вимоги по збереженню інформації при аваріях

При виникненні аварійних сиуацій програмний код та файли налаштувань повинні зберігатися на MicroSD-карті, а їх резервна копія – на зовнішньому носії (ПК, хмарне сховище).

3.4.2 Вимоги по стандартизації і уніфікації

Система повинна відповідати вимогам ергономіки і зручності користування. Апаратна частина повинна базуватися на поширених та стандартизованих компонентах та інтерфейсах (USB, GPIO, CSI, 3.5 мм аудіо).

3.4.3 Вимоги до функцій (завдань), що виконуються системою:

– виявлення перешкод на відстані до 4 метрів;
– голосове сповіщення про тип та відстань об’єкта;
– автономна робота протягом не менше 8 годин;
– тактильне та звукове сповіщення про критичну небезпеку.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

а) Структурна схема системи.

б) Схема електрична принципова.

в) Блок-схема алгоритму роботи.

г) Блок-схема алгоритму роботи програмного забезпечення.

*Примітка: У комплект документації можуть вноситися зміни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ з/п	Назва етапів роботи	Термін виконання етапів роботи
1.	*Розробка технічного завдання*	*27.01.. – 02.02. р.*
2.	*Аналіз технічного завдання*	*27.01. – 02.02.*
3.	*Аналіз технологій розробки системи голосового інформування*	*03.02. – 05.03.*
4.	*Обґрунтування вибору апаратного забезпечення*	*06.03. – 25.03.*
5.	*Реалізація програмного забезпечення системи*	*26.03. – 16.04.*
6.	*Реалізація апаратного забезпечення системи*	*17.04. – 10.05.*
7.	*Безпека життєдіяльності, основи охорони праці*	*11.05. – 01.06.*
8.	*Оформлення пояснювальної записки і графічного матеріалу*	*2.06.. – 13.06.*
9.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*9.06. – 15.06.*
10.	*Попередній захист кваліфікаційної роботи бакалавра*	*16.06. – 22.06.*
11.	*Захист кваліфікаційної роботи бакалавра*	*26.06.*

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

Перелік елементів

Поз. позначення	*Найменування*	*Кіл.*	*Примітка*
	Мікрокомп'ютер та модулі		
U1	Raspberry Pi 3 Model B	1	1.2ГГц, 1ГБ RAM
U2	HC-SR04	1	
U3	Waveshare RPi Camera (G)	1	5МП, 160° FOV
	Пристрої виведення		
LS1	Зумер активний	1	5В
M1	Вібромотор дисковий	1	3-5В
U4	Активні колонки	2	Підключаються до 3.5мм аудіовиходу
	Інші компоненти		
J1	CSI	1	
J2	USB_TypeA	1	
-	Набір з'єднувальних дротів	1	Dupont, F-F
-	Кабель 3.5мм - 3.5мм (AUX)	1	Для аудіосистеми

Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	*КС КРБ 123.212.00.00 ПЕ1*		
Розроб.		*Конфедерат О.О.*			*Комп'ютеризована система голосового інформування людей з вадами зору на базі Raspberry PI Перелік елементів*	*Літ.*	*Арк.* / *Аркушів*
Перевір.		*Стадник Н.Б.*					
Реценз.		*Марценко С.В.*				*ТНТУ, каф. КС, гр. СІ-41*	
Н. Контр.		*Луцик Н.С.*					
Затверд.		*Осухівська Г.М.*					

Додаток В

Лістинг програмного коду

```
import cv2
import numpy as np
import time
import RPi.GPIO as GPIO
import pyttsx3
from threading import Thread, Lock
from queue import Queue
import logging
import os

# Налаштування GPIO
GPIO_TRIGGER = 20
GPIO_ECHO = 21
GPIO_VIBRATOR = 18
GPIO_BUZZER = 17

# Константи відстаней
DISTANCE_CRITICAL_CM = 75
DISTANCE_WARN_CM = 200
MAX_DETECTION_DISTANCE_CM = 400

# Налаштування моделі
MODEL_PATH = 'detect.tflite'
LABEL_PATH = 'labelmap.txt'
MIN_CONFIDENCE = 0.60

# Налаштування логування
log_level = os.getenv('LOG_LEVEL', 'INFO').upper()
logging.basicConfig(
    level=getattr(logging, log_level, logging.INFO),
    format='%(asctime)s - %(levelname)s - %(message)s'
)
logger = logging.getLogger(__name__)

class SystemState:
    def __init__(self):
        self.distance = float('inf')
        self.detected_object = None
        self.camera_active = True
        self.ultrasonic_active = True
        self.emergency_mode = False
        self.lock = Lock()

    def update_distance(self, distance):
        with self.lock:
            self.distance = distance

    def update_object(self, obj):
```

```
        with self.lock:
            self.detected_object = obj

    def set_emergency_mode(self, mode):
        with self.lock:
            self.emergency_mode = mode
            logger.warning(f"Аварійний режим: {'Активовано' if
mode else 'Деактивовано'}")

class VoiceAssistant:
    def __init__(self):
        self.engine = None
        self.lock = Lock()
        self.last_spoken_text = ""
        self.last_spoken_time = 0
        self.speech_cooldown = 3
        self._initialize_engine()

    def _initialize_engine(self):
        try:
            self.engine = pyttsx3.init()
            self.engine.setProperty('rate', 170)
            voices = self.engine.getProperty('voices')
            for voice in voices:
                if voice and ('uk' in voice.id.lower() or
'ukrainian' in voice.name.lower()):
                    self.engine.setProperty('voice', voice.id)
                    logger.info(f"Знайдено український голос:
{voice.name}")
                    break
        except Exception as e:
            logger.error(f"Помилка ініціалізації голосового
движка: {e}")
            self.engine = None

    def speak(self, text, force=False):
        if not self.engine:
            logger.warning(f"[VOICE OFF] {text}")
            return
        current_time = time.time()
        if not force and text == self.last_spoken_text and
(current_time - self.last_spoken_time) < self.speech_cooldown:
            return
        with self.lock:
            try:
                logger.info(f"[VOICE] Озвучення: '{text}'")
                self.last_spoken_text = text
                self.last_spoken_time = current_time
                self.engine.say(text)
                self.engine.runAndWait()
            except Exception as e:
                logger.error(f"Помилка озвучення: {e}")
```

```
class UltrasonicSensor:
    def __init__(self, system_state, alert_queue):
        self.system_state = system_state
        self.alert_queue = alert_queue
        self.running = True

    def measure_distance(self):
        try:
            GPIO.output(GPIO_TRIGGER, True)
            time.sleep(0.00001)
            GPIO.output(GPIO_TRIGGER, False)
            timeout_start = time.time()
            timeout_duration = 0.5

            while GPIO.input(GPIO_ECHO) == 0:
                start_time = time.time()
                if (time.time() - timeout_start) >
timeout_duration:
                    return None

            while GPIO.input(GPIO_ECHO) == 1:
                stop_time = time.time()
                if (time.time() - timeout_start) >
timeout_duration:
                    return None

            time_elapsed = stop_time - start_time
            distance = (time_elapsed * 34300) / 2
            if 2 <= distance <= MAX_DETECTION_DISTANCE_CM:
                return distance
            return None
        except Exception as e:
            logger.debug(f"Помилка вимірювання відстані: {e}")
            return None

    def worker(self):
        consecutive_errors = 0
        max_errors = 5
        while self.running:
            try:
                distance = self.measure_distance()
                if distance is not None:
                    consecutive_errors = 0
                    self.system_state.update_distance(distance)
                    self.alert_queue.put({'type': 'distance',
'value': distance})
                else:
                    consecutive_errors += 1
                    if consecutive_errors >= max_errors:
                        self.system_state.ultrasonic_active =
False

self.system_state.set_emergency_mode(True)
```

```
                        self.alert_queue.put({'type': 'error',
'value': 'Відмова ультразвукового датчика'})
                        break
                time.sleep(0.2)
            except Exception as e:
                logger.error(f"Помилка в роботі ультразвукового
датчика: {e}")
                self.system_state.ultrasonic_active = False
                self.system_state.set_emergency_mode(True)
                break

class CameraDetector:
    def __init__(self, system_state, alert_queue):
        self.system_state = system_state
        self.alert_queue = alert_queue
        self.running = True
        self.interpreter = None
        self.labels = []
        self.camera = None

    def initialize(self):
        try:
            from tflite_runtime.interpreter import Interpreter
            self.interpreter =
Interpreter(model_path=MODEL_PATH)
            self.interpreter.allocate_tensors()

            with open(LABEL_PATH, 'r', encoding='utf-8') as f:
                self.labels = [line.strip() for line in
f.readlines()]

            self.camera = cv2.VideoCapture(0)
            if not self.camera.isOpened():
                raise Exception("Не вдалося відкрити камеру")

            self.camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
            self.camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
            return True
        except Exception as e:
            logger.error(f"Помилка ініціалізації камери: {e}")
            return False

    def detect_objects(self, frame):
        try:
            input_details = self.interpreter.get_input_details()
            output_details =
self.interpreter.get_output_details()
            height, width = input_details[0]['shape'][1],
input_details[0]['shape'][2]

            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame_resized = cv2.resize(frame_rgb, (width,
height))
```

```
            input_data = np.expand_dims(frame_resized, axis=0)

            if input_details[0]['dtype'] == np.float32:
                input_data = (np.float32(input_data) - 127.5) /
127.5


self.interpreter.set_tensor(input_details[0]['index'],
input_data)
            self.interpreter.invoke()

            scores =
self.interpreter.get_tensor(output_details[2]['index'])[0]
            classes =
self.interpreter.get_tensor(output_details[1]['index'])[0]

            if len(scores) > 0 and scores[0] > MIN_CONFIDENCE:
                best_class_id = int(classes[0])
                return self.labels[best_class_id] if
best_class_id < len(self.labels) else None
            return None
        except Exception as e:
            logger.error(f"Помилка розпізнавання: {e}")
            return None

    def worker(self):
        if not self.initialize():
            self.system_state.camera_active = False
            self.system_state.set_emergency_mode(True)
            self.alert_queue.put({'type': 'error', 'value':
'Відмова камери'})
            return

        while self.running:
            try:
                ret, frame = self.camera.read()
                if not ret:
                    continue

                detected_object = self.detect_objects(frame)
                self.system_state.update_object(detected_object)
                self.alert_queue.put({'type': 'object', 'value':
detected_object})
                time.sleep(0.1)
            except Exception as e:
                logger.error(f"Критична помилка камери: {e}")
                break

        if self.camera:
            self.camera.release()

class AlertSystem:
    def __init__(self):
```

```
        self.vibrator_active = False
        self.buzzer_active = False

    def activate_critical_alert(self):
        GPIO.output(GPIO_VIBRATOR, True)
        GPIO.output(GPIO_BUZZER, True)
        self.vibrator_active = True
        self.buzzer_active = True

    def deactivate_alert(self):
        GPIO.output(GPIO_VIBRATOR, False)
        GPIO.output(GPIO_BUZZER, False)
        self.vibrator_active = False
        self.buzzer_active = False

def generate_voice_message(system_state):
    with system_state.lock:
        distance = system_state.distance
        detected_object = system_state.detected_object

    message_parts = []

    if detected_object:
        message_parts.append(f"Попереду {detected_object}")
        if distance < DISTANCE_WARN_CM:
            message_parts.append(f"на відстані {int(distance)}
сантиметрів")
    elif distance < DISTANCE_WARN_CM:
        message_parts.append("Попереду перешкода")
        message_parts.append(f"на відстані {int(distance)}
сантиметрів")

    return ". ".join(message_parts) if message_parts else None

def main():
    # Налаштування GPIO
    GPIO.setmode(GPIO.BCM)
    GPIO.setup([GPIO_TRIGGER, GPIO_VIBRATOR, GPIO_BUZZER],
GPIO.OUT)
    GPIO.setup(GPIO_ECHO, GPIO.IN)
    GPIO.output([GPIO_VIBRATOR, GPIO_BUZZER], GPIO.LOW)

    # Ініціалізація компонентів системи
    system_state = SystemState()
    alert_queue = Queue()
    voice = VoiceAssistant()
    alert_system = AlertSystem()

    # Створення сенсорів
    ultrasonic = UltrasonicSensor(system_state, alert_queue)
    camera = CameraDetector(system_state, alert_queue)

    # Запуск потоків
```

```
    Thread(target=ultrasonic.worker, daemon=True).start()
    Thread(target=camera.worker, daemon=True).start()

    # Початкове повідомлення
    voice.speak("Система навігації запущена", force=True)

    last_message_time = 0

    try:
        while True:
            try:
                alert = alert_queue.get(timeout=0.5)
            except:
                continue

            current_time = time.time()

            # Обробка помилок системи
            if alert['type'] == 'error':
                voice.speak(f"Системна помилка:
{alert['value']}", force=True)
                logger.error(alert['value'])
                continue

            # Критична відстань - негайне попередження
            if system_state.ultrasonic_active and
system_state.distance < DISTANCE_CRITICAL_CM:
                alert_system.activate_critical_alert()
                voice.speak("Увага! Критична відстань!",
force=True)
                time.sleep(0.5)
                alert_system.deactivate_alert()
                continue
            else:
                alert_system.deactivate_alert()

            # Регулярні голосові повідомлення
            if current_time - last_message_time >= 2:
                message = generate_voice_message(system_state)
                if message:
                    voice.speak(message)
                    last_message_time = current_time

    except KeyboardInterrupt:
        logger.info("Зупинка системи користувачем")
    finally:
        # Зупинка всіх компонентів
        ultrasonic.running = False
        camera.running = False
        alert_system.deactivate_alert()
        GPIO.cleanup()
        logger.info("Система зупинена")
```

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