Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» червня 2025 р.

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Модель інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів»

Спеціальність 151 –Автоматизація та комп'ютерно-інтегровані технології
Галузь знань 15 – Автоматизація та приладобудування
Освітня програма «Автоматизація та комп'ютерно-інтегровані технології»

Захищено на засіданні
Екзаменаційної комісії № 46
протокол № __ від __.06.2025 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **ЧУГАЙ А.М.**

Виконав:
Студент групи КУ– 41
НАЛІМОВ Богдан Олександрович
_____

Керівник: старший викладач кафедри комп'ютерних систем та робототехніки
КОПЄЙКІН Володимир Васильович
_____

Рецензент: к.ф.-м.н., доцент, доцент зво кафедри математичного моделювання та аналізу даних
ПОКЛОНСЬКИЙ Євген Васильович
_____

Харків – 2025

АНОТАЦІЯ

Надана кваліфікаційна робота бакалавра присвячена розробці інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів. Загальний обсяг роботи складає 64 сторінки, із яких 39 сторінок основної частини з 1 рисунком, 17 найменуваннями списку використаних джерел та чотирьма додатками.

Метою кваліфікаційної роботи є розробка моделі інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів для забезпечення автономного переміщення з уникненням перешкод і розпізнаванням об’єктів за кольором у реальному часі.

Об’єкт дослідження – роботизований IoT-пристрій, оснащений сенсорами відстані (ультразвуковими та інфрачервоними) та камерою ESP32-CAM, що функціонує в рамках концепції Інтернету речей (IoT).

Предмет дослідження – методи та алгоритми інтелектуального управління роботизованим IoT-пристроєм, які базуються на обробці та аналізі даних із сенсорів відстані та камери для забезпечення автономної навігації та розпізнавання кольорів.

Проблема, яка вирішується в кваліфікаційній роботі, полягає в забезпеченні надійного, ефективного та безпечного автономного переміщення робота в невідомому або частково відомому середовищі шляхом інтеграції даних із різнорідних сенсорів, обробки зображень у реальному часі та оптимізації обчислювальних ресурсів для бюджетного IoT-пристрою.

Область застосування – автоматизація логістичних процесів, освіта, побутові системи моніторингу та прибирання. Розроблена система може використовуватися для автоматизації доставки вантажів, як навчальний інструмент для вивчення IoT і робототехніки, а також у системах патрулювання приміщень.

КЛЮЧОВІ СЛОВА: РОБОТИЗОВАНЕ ШАСІ, IoT, АНАЛІЗ ДАНИХ, СЕНСОР, ARDUINO UNO, ESP32-CAM, OPENCV, РЕАКТИВНЕ УПРАВЛІННЯ, РОЗПІЗНАВАННЯ КОЛЬОРІВ, АВТОНОМНА НАВІГАЦІЯ.

ABSTRACT

The bachelor's qualification work is dedicated to the development of a model of an intelligent control system for a robotic IoT device based on sensor data analysis. The total volume of the work is 64 pages, of which 39 pages are the main part, including 1 figure, 17 references in the list of sources used, and four appendices.

The objective of the qualification work is to develop an intelligent control system for a robotic IoT device based on sensor data analysis to enable autonomous movement with obstacle avoidance and real-time color-based object recognition.

The object of the study is a robotic IoT device equipped with distance sensors (ultrasonic and infrared) and an ESP32-CAM camera, operating within the framework of the Internet of Things (IoT) concept.

The subject of the study is the methods and algorithms for intelligent control of a robotic IoT device, based on the processing and analysis of data from distance sensors and a camera to ensure autonomous navigation and color recognition.

The problem addressed in the qualification work is to ensure reliable, efficient, and safe autonomous movement of the robot in an unknown or partially known environment by integrating data from heterogeneous sensors, processing images in real time, and optimizing computational resources for a cost-effective IoT device.

The scope of application includes automation of logistics processes, education, and household monitoring and cleaning systems. The developed system can be used for automating cargo delivery, as an educational tool for studying IoT and robotics, and in room patrolling systems.

KEY WORDS: ROBOTIC CHASSIS, IoT, DATA ANALYSIS, SENSOR, ARDUINO UNO, ESP32-CAM, OPENCV, REACTIVE CONTROL, COLOR RECOGNITION, AUTONOMOUS NAVIGATION.

ЗМІСТ

ВСТУП

У зв’язку зі стрімким розвитком технологій Інтернету речей (IoT) та робототехніки, все більш актуальною стає проблема створення ефективних систем управління роботизованими пристроями, які здатні автономно взаємодіяти з навколишнім середовищем. Такі системи відіграють ключову роль у автоматизації процесів у різних сферах, включаючи логістику, освіту та побутові задачі.

Актуальність роботи. Однією з ключових проблем у розвитку роботизованих IoT-пристроїв є забезпечення їхньої здатності до автономної навігації та обробки даних із різнорідних сенсорів у реальному часі. Складність інтеграції даних від ультразвукових, інфрачервоних сенсорів та камер, а також обмеження обчислювальних ресурсів бюджетних мікроконтролерів, таких як Arduino UNO та ESP32-CAM, створюють виклики для створення надійних і енергоефективних систем. Крім того, точність розпізнавання об’єктів за кольором залежить від умов освітлення, а ефективність навігації обмежується відсутністю планування маршрутів у складних середовищах. Таким чином, розробка інтелектуальної системи управління, яка оптимізує обробку даних і забезпечує автономність, є актуальною задачею з високим практичним потенціалом.

Метою дослідження є розробка моделі інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів для забезпечення автономного переміщення з уникненням перешкод і розпізнаванням об’єктів за кольором у реальному часі.

Об’єкт дослідження – роботизований IoT-пристрій, оснащений сенсорами відстані (ультразвуковими та інфрачервоними) та камерою ESP32-CAM, що функціонує в рамках концепції Інтернету речей (IoT).

Методи дослідження: методи математичного моделювання, обробки даних сенсорів, аналізу зображень, реактивного управління, збір даних, а також методи інтеграції даних із різнорідних джерел.

Предмет дослідження – методи та алгоритми інтелектуального управління роботизованим IoT-пристроєм, які базуються на обробці та аналізі даних із сенсорів відстані та камери для забезпечення автономної навігації та розпізнавання кольорів.

Завдання дослідження

1. Провести аналіз існуючих методів і алгоритмів обробки даних із сенсорів відстані та камер у контексті управління роботизованими IoT-пристроями.

2. Розробити алгоритм інтеграції даних із ультразвукових, інфрачервоних сенсорів та камери ESP32-CAM для створення єдиної моделі оточення.

3. Реалізувати систему розпізнавання кольорів об’єктів за допомогою камери ESP32-CAM, адаптовану до умов змінного освітлення.

4. Розробити алгоритм автономної навігації та уникнення перешкод, який використовує комбіновані дані сенсорів для прийняття рішень у реальному часі.

5. Провести експериментальне тестування розробленої системи на реальному роботизованому шасі, оцінити її ефективність і визначити напрями вдосконалення.

РОЗДІЛ 1.
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Сучасний розвиток технологій Інтернету речей (IoT) і робототехніки характеризується швидким прогресом у створенні автономних систем, які здатні взаємодіяти з навколишнім середовищем без постійного втручання людини. IoT-пристрої, оснащені сенсорами, мікроконтролерами та можливостями мережевого зв'язку, знаходять застосування в різних сферах, таких як логістика, промислове виробництво, сільське господарство, медицина та побутові системи. Робототехніка, у свою чергу, інтегрує ці технології для створення мобільних платформ, здатних виконувати складні задачі, такі як автономна навігація, розпізнавання об'єктів і обробка даних у реальному часі.

1.1 Огляд сучасного стану розвитку IoT і робототехніки

Тенденції розвитку IoT і робототехніки:

1. Мініатюризація та доступність апаратного забезпечення: Розвиток бюджетних мікроконтролерів, таких як Arduino UNO та ESP32, а також модулів із вбудованими камерами (наприклад, ESP32-CAM), дозволяє створювати економічно вигідні рішення для малих і середніх проектів. Ці платформи забезпечують достатню обчислювальну потужність для базової обробки даних і зв'язку через Wi-Fi або Bluetooth, що робить їх популярними в освітніх і дослідницьких проектах.

2. Інтеграція сенсорних технологій: Сучасні роботизовані IoT-пристрої використовують комбінацію сенсорів, таких як ультразвукові, інфрачервоні та камери, для створення повноцінної моделі оточення. Дослідження [9, 10] підкреслюють, що злиття даних із різнорідних сенсорів підвищує точність і надійність систем, хоча вимагає складних алгоритмів інтеграції.

3. Розвиток алгоритмів обробки даних: Методи машинного навчання, зокрема нейронні мережі та фільтри Калмана, активно застосовуються для

обробки сенсорних даних і розпізнавання об’єктів. Наприклад, бібліотека OpenCV широко використовується для аналізу зображень у реальному часі, що підтверджується дослідженнями [13]. Однак такі методи часто потребують значних обчислювальних ресурсів, що є викликом для бюджетних IoT-пристроїв. У праці [6, 8] зазначається, що використання методів машинного навчання, таких як нейронні мережі, дозволяє створювати адаптивні моделі розпізнавання, які менш вимогливі до ресурсів, що робить їх придатними для бюджетних IoT-пристроїв.

4. Автономна навігація та уникнення перешкод: Реактивні методи, такі як алгоритм "potential fields" [11, 14], і планові методи, наприклад, алгоритм A* [12, 15], активно розвиваються для забезпечення ефективної навігації в невідомих або динамічних середовищах. Поєднання цих підходів із технологіями IoT, такими як ESP32-CAM, дозволяє створювати системи, здатні обробляти дані в реальному часі та приймати швидкі рішення щодо руху.

5. Енергоефективність та безпека: Розробка енергоефективних алгоритмів і технологій шифрування даних сприяє створенню безпечних і довговічних автономних систем, що є критично важливим для їхнього широкого застосування.

Цей огляд демонструє актуальність і важливість інтеграції IoT і робототехніки для створення економічно ефективних, автономних систем, які можуть бути використані в логістиці, освіті, побуті та інших сферах. Подальший розвиток у цій галузі передбачає вдосконалення алгоритмів машинного навчання та планування маршрутів для підвищення ефективності й автономності таких систем.

1.2 Огляд та аналіз публікацій за темою

Сучасні дослідження в галузі робототехніки та Інтернету речей (IoT) активно зосереджені на створенні автономних систем, здатних ефективно

взаємодіяти з оточенням без постійного контролю з боку людини. Ключову роль у таких системах відіграють сенсори, які забезпечують збір даних про навколишнє середовище, та алгоритми, що обробляють ці дані для прийняття рішень. У цьому підрозділі розглядаються основні підходи до використання сенсорів відстані, камер для розпізнавання кольорів, інтеграції даних і автономної навігації, що є актуальними для теми даної роботи.

Сенсори відстані

Сенсори відстані є фундаментальною складовою роботизованих систем, оскільки дозволяють визначати розташування об’єктів у просторі та уникати зіткнень. Найпоширенішими типами таких сенсорів є ультразвукові та інфрачервоні.

Ультразвукові сенсори функціонують шляхом вимірювання часу, за який звукова хвиля відбивається від об’єкта і повертається до джерела. Їхня популярність зумовлена простотою конструкції, низькою вартістю та достатньою ефективністю в більшості застосувань . Дослідження [9] демонструє, що використання мікроконтролерів дозволяє підвищити точність вимірювань до ±1 см, що підтверджує доцільність застосування ультразвукових сенсорів у цій роботі. Однак ці сенсори мають обмеження, такі як чутливість до акустичного шуму та знижена точність при роботі з об’єктами неправильної форми.

Інфрачервоні сенсори визначають відстань на основі інтенсивності відбитого світла, що забезпечує високу точність (±1 см) на коротких відстанях, зазвичай до 80 см (що висвітлено у праці [10]). Вони забезпечують вищу точність у порівнянні з ультразвуковими сенсорами, але їхня робота залежить від умов освітлення та поверхневих характеристик об’єктів, таких як колір чи текстура.

У даній роботі використовується комбінація обох типів сенсорів, що дозволяє компенсувати їхні окремі недоліки. Дослідження демонструє, що

злиття даних із ультразвукових та інфрачервоних сенсорів підвищує надійність і точність вимірювань, що є важливим для автономної навігації в реальних умовах.

Камера ESP32 CAM для розпізнавання кольорів

Камера ESP32 CAM є економічно вигідним і компактним рішенням для IoT-пристроїв, яке інтегрується з мікроконтролером ESP32. Вона широко застосовується для задач обробки зображень, зокрема для розпізнавання кольорів, що є критично важливим для ідентифікації об’єктів у робототехніці. Розпізнавання кольорів за допомогою камери ESP32 CAM базується на аналізі зображень у колірному просторі HSV, що дозволяє ідентифікувати об’єкти за їхніми колірними характеристиками. Дослідження [13] показує, що комбінація колірних і формових ознак підвищує точність виявлення об’єктів у напівструктурованих середовищах, що підтверджує доцільність використання такого підходу в цій роботі.

Існує кілька підходів до розпізнавання кольорів:

Порогове розпізнавання базується на визначенні діапазонів значень у колірних просторах, таких як RGB або HSV. Цей метод є простим у реалізації та не вимагає значних обчислювальних ресурсів, але його ефективність знижується при зміні умов освітлення.

Методи машинного навчання, наприклад, нейронні мережі, дозволяють створювати більш адаптивні моделі розпізнавання, враховуючи шум і варіації в даних (що добре висвітлено у роботах [6] та [8]). Однак такі методи потребують значних обчислювальних ресурсів і попереднього навчання на великих наборах даних.

У контексті цієї роботи камера ESP32 CAM використовуватиметься для ідентифікації кольорових об’єктів у реальному часі. Дослідження [6] показує успішне застосування подібних камер для розпізнавання кольорових міток у

системах автономної навігації, що підтверджує доцільність її використання.

Інтеграція даних із різних сенсорів

Для створення цілісної моделі оточення необхідна інтеграція даних із сенсорів відстані та камери. Це дозволяє підвищити точність сприйняття середовища та компенсувати обмеження окремих джерел інформації. Серед основних методів інтеграції даних виділяють:

- Фільтр Калмана, який широко застосовується для оцінки стану системи на основі зашумлених даних і прогнозування її поведінки.
- Байєсівські мережі, що моделюють невизначеність і залежності між джерелами даних, забезпечуючи ймовірнісний підхід до прийняття рішень [8, 16].
- Методи машинного навчання, такі як глибокі нейронні мережі, які здатні автоматично виявляти закономірності в даних із різних сенсорів.

У цій роботі було розроблено алгоритм інтеграції, який враховує специфіку роботизованого IoT-пристрою та комбінує дані для забезпечення стабільної роботи в динамічному середовищі.

Алгоритми автономної навігації та уникнення перешкод

Автономна навігація є ключовою задачею для роботів, що функціонують у невідомих або змінних умовах. Існують різні підходи до її реалізації:

- Реактивні методи, такі як алгоритм "potential fields" (висвітлений у працях [11,14]), де перешкоди створюють відштовхуючі сили, а цілі — притягуючі, дозволяють швидко реагувати на зміни в оточенні.
- Планові методи, наприклад, алгоритм A* (висвітлений у працях [12], [15]), будують оптимальний шлях на основі карти оточення.
- Методи навчання з підкріпленням дають змогу роботу адаптуватися до середовища через проби та помилки (зазначений у праці [7]).

Для даного дослідження буде розроблено гібридний алгоритм, що поєднує реактивні та планові підходи, використовуючи комбіновані дані сенсорів для ефективної навігації та уникнення перешкод.

Висновки до розділу 1

Аналіз літератури свідчить про наявність широкого спектра методів і технологій для обробки даних із сенсорів, їх інтеграції та реалізації автономної навігації. Однак більшість сучасних рішень зосереджені на окремих аспектах, таких як використання одного типу сенсорів чи специфічних алгоритмів. Це підкреслює потребу в розробці комплексного підходу, який би враховував особливості роботизованих IoT-пристроїв із комбінацією ультразвукових, інфрачервоних сенсорів і камери ESP32 CAM.

Розробка інтелектуальної системи управління для такого пристрою є актуальною задачею, що має практичну цінність у сферах автоматизації, логістики та моніторингу. У наступних розділах буде представлено детальний опис розробки алгоритмів, їхньої реалізації та результатів експериментального тестування.

РОЗДІЛ 2.
РОЗРОБКА МОДЕЛІ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ УПРАВЛІННЯ

Розробка моделі інтелектуальної системи управління для роботизованого IoT-пристрою є складним завданням, яке вимагає ретельного аналізу на початковому етапі. Аналіз розв'язуваної задачі є фундаментом для подальшої роботи, оскільки він дозволяє чітко визначити цілі системи, сформулювати вимоги до її функціонування та врахувати обмеження, що впливають на процес розробки й експлуатації. Далі буде детально описано проблему, яку вирішує система, вимоги до неї (функціональні та нефункціональні), а також обмеження та умови роботи, що формують контекст проекту.

2.1 Опис проблеми, яку вирішує система

Роботизований IoT-пристрій, що розробляється в рамках цієї бакалаврської роботи, призначений для автономного переміщення в невідомому або частково відомому середовищі з виконанням двох ключових функцій: уникнення перешкод і розпізнавання об'єктів за кольором у реальному часі. Основна проблема, яку вирішує інтелектуальна система управління, полягає в забезпеченні надійного, ефективного та безпечного руху пристрою на основі даних, отриманих із різноманітних сенсорів — зокрема, ультразвукових та інфрачервоних сенсорів відстані, а також камери ESP32 CAM. Конкретніше, система має вирішувати такі задачі:

1. Автономне переміщення: Пристрій повинен самостійно рухатися в просторі, не потребуючи зовнішнього управління, і адаптуватися до змін у навколишньому середовищі.
2. Уникнення перешкод: Система має виявляти перешкоди на шляху руху та приймати рішення щодо їх об'їзду, забезпечуючи безпеку пристрою та оточення.

3. Розпізнавання об’єктів за кольором: Використовуючи камеру, система повинна ідентифікувати об’єкти за їх кольором для виконання специфічних завдань, наприклад, знаходження цілей, міток або орієнтирів у просторі.

Ця проблема є актуальною в сучасному світі, де автономні роботизовані системи та IoT-технології набувають дедалі більшого поширення. Такі пристрої можуть застосовуватися в логістиці (наприклад, для доставки вантажів у складських приміщеннях), моніторингу інфраструктури (огляд важкодоступних зон) або навіть у побутових задачах (розумні домашні роботи). Розв’язання цієї задачі вимагає інтеграції апаратного забезпечення, обробки даних у реальному часі та інтелектуальних алгоритмів управління.

Вимоги до системи

Для успішної реалізації поставленої задачі система повинна відповідати чітко визначеним вимогам, які поділяються на функціональні (що система має робити) та нефункціональні (якими характеристиками вона має володіти).

Функціональні вимоги

1. Збір даних із сенсорів: Система має забезпечувати стабільне зчитування даних із ультразвукових та інфрачервоних сенсорів відстані, а також із камери ESP32 CAM для отримання інформації про навколишнє середовище.

2. Обробка даних: Необхідно реалізувати алгоритми для фільтрації шумів у сенсорних даних, інтеграції інформації з різних джерел і розпізнавання кольорів об’єктів на зображеннях із камери.

3. Уникнення перешкод: Система повинна в реальному часі виявляти перешкоди, аналізувати їх розташування та змінювати траєкторію руху для їх об’їзду.

4. Розпізнавання об’єктів: Використовуючи дані з камери, система має точно ідентифікувати об’єкти за кольором і застосовувати цю інформацію для навігації чи виконання завдань.

5. Керування рухом: На основі оброблених даних система повинна генерувати команди для двигунів, забезпечуючи плавне, безпечне та цілеспрямоване переміщення пристрою.

Нефункціональні вимоги

1. Швидкодія: Обробка даних і прийняття рішень мають відбуватися в реальному часі з затримкою не більше 100 мс для критичних операцій, таких як уникнення перешкод.

2. Точність: Точність вимірювання відстані ультразвуковими сенсорами має становити ±2 см, інфрачервоними — ±1 см. Розпізнавання кольорів повинно бути точним у 95% випадків за стабільного освітлення.

3. Енергоефективність: Оскільки пристрій живиться від батареї, система має мінімізувати споживання енергії шляхом оптимізації алгоритмів і використання енергоощадних режимів роботи компонентів.

4. Надійність: Система повинна бути стійкою до збоїв, із можливістю автоматичного відновлення після помилок або перезавантаження.

5. Масштабованість: Архітектура системи має бути гнучкою, дозволяючи додавати нові сенсори чи функціональні модулі без значних змін у базовому коді.

Ці вимоги формують основу для оцінки ефективності системи та визначають критерії її успішної реалізації.

Обмеження та умови роботи

Розробка системи відбувається в умовах певних обмежень, які впливають на вибір технічних рішень, алгоритмів і підходів до реалізації. Ці обмеження

поділяються на технічні, середовищні та ресурсні.

Технічні обмеження

1. Обмежена обчислювальна потужність ESP32: Мікроконтролер ESP32, який є основою системи, має обмежені ресурси (520 КБ SRAM, двоядерний процесор із частотою до 240 МГц), що вимагає ретельної оптимізації коду та алгоритмів для забезпечення швидкодії.

2. Обмежена пропускна здатність мережі: Як IoT-пристрій, система може передавати дані через Wi-Fi, але обмежена пропускна здатність і можливі затримки в мережі впливають на швидкість обміну інформацією.

3. Обмежена роздільна здатність камери: Камера ESP32 CAM підтримує роздільну здатність до 1600x1200 пікселів, що достатньо для базового розпізнавання кольорів, але може бути недостатнім для складних сцен із великою кількістю деталей.

Середовищні обмеження

1. Змінне освітлення: Робота камери та інфрачервоних сенсорів залежить від умов освітлення, що може знижувати точність розпізнавання кольорів і вимірювання відстані в умовах недостатнього або надмірного світла.

2. Акустичний шум: Ультразвукові сенсори чутливі до зовнішнього шуму, що вимагає додаткової фільтрації даних для забезпечення їхньої достовірності.

3. Фізичні перешкоди: Пристрій має функціонувати в середовищі з різноманітними перешкодами (стіни, меблі, рухомі об’єкти), що вимагає адаптивних алгоритмів навігації.

Ресурсні обмеження

1. Бюджет: Розробка ведеться з використанням доступних і недорогих компонентів (ESP32, стандартні сенсори), що виключає можливість використання більш просунутих, але дорогих рішень.

2. Час: Обмежений час на розробку й тестування системи вимагає чіткого планування та пріоритизації завдань.

3. Людські ресурси: Оскільки проект виконується одним розробником за допомогою научного керівника, необхідно оптимально розподіляти зусилля між аналізом, програмуванням і тестуванням.

Аналіз розв’язуваної задачі дозволив чітко сформулювати проблему, яку має вирішити інтелектуальна система управління роботизованим IoT-пристроєм: забезпечення автономного переміщення з уникненням перешкод і розпізнаванням об’єктів за кольором у реальному часі. Було визначено функціональні вимоги, що охоплюють збір і обробку даних, керування рухом і розпізнавання об’єктів, а також нефункціональні вимоги, такі як швидкодія, точність, енергоефективність і надійність. Крім того, описано технічні, середовищні та ресурсні обмеження, які необхідно враховувати під час розробки. Цей аналіз створює міцну основу для подальшого вибору компонентів, розробки алгоритмів і тестування системи, що буде розглянуто в наступних підпунктах роботи.

2.2 Огляд компонентів системи

Успішна розробка інтелектуальної системи управління для роботизованого IoT-пристрою значною мірою залежить від ретельного вибору апаратних компонентів та їхньої ефективної інтеграції. Кожен компонент системи виконує специфічну функцію, а їхні технічні характеристики та можливості визначають продуктивність, надійність і здатність пристрою виконувати поставлені задачі. У цьому підпункті буде детально розглянуто

основні компоненти системи: мікроконтролер ESP32, ультразвукові сенсори та інфрачервоні сенсори. Для кожного компонента буде описано технічні характеристики, можливості, переваги використання та їхня роль у реалізації проекту.

Arduino UNO

Технічні характеристики (зазначені у праці [1]): Arduino UNO — це мікроконтролер на базі чіпа ATmega328P, який є центральним елементом системи завдяки своїм можливостям:

- Процесор: 8-бітний AVR з тактовою частотою 16 МГц, що забезпечує достатню обчислювальну потужність для базового керування.
- Пам’ять: 2 КБ SRAM, 32 КБ флеш-пам’яті для коду та 1 КБ EEPROM для даних.
- Інтерфейси: 14 цифрових і 6 аналогових пінів, підтримка UART, I2C і SPI для зв’язку з іншими пристроями.

Роль у системі:

Arduino UNO є центральним контролером, який:

- Отримує дані від ультразвукових та інфрачервоних сенсорів для оцінки відстані до перешкод.
- Приймає результати аналізу зображень від ESP32-CAM.
- Приймає рішення щодо руху робота (наприклад, поворот, зупинка чи об’їзд перешкод).

Переваги:

- Простота: Легке програмування через Arduino IDE.
- Гнучкість: Можливість підключення до сенсорів і драйверів двигунів.
- Доступність: Низька вартість і широка підтримка спільноти.

У цій системі Arduino UNO обробляє всю логіку прийняття рішень, використовуючи дані від сенсорів і результати аналізу зображень, отримані від ESP32-CAM.

ESP32-CAM

Технічні характеристики (зазначені у праці [2]): ESP32-CAM — це модуль із вбудованою камерою та мікроконтролером ESP32, призначений для роботи з зображеннями:

- Процесор: Двоядерний 32-бітний Xtensa LX6 із частотою до 240 МГц.
- Пам’ять: 520 КБ SRAM, підтримка зовнішньої флеш-пам’яті до 4 МБ.
- Камера: OV2640 з роздільною здатністю до 2 МП.
- Зв’язок: Вбудований Wi-Fi для бездротової передачі даних.

Роль у системі:

ESP32-CAM виконує обмежені, але важливі функції:

- Захоплює зображення за допомогою камери.
- Надсилає зображення на віддалений сервер для аналізу (наприклад, розпізнавання кольору чи об’єктів).
- Отримує результати аналізу від сервера та передає їх на Arduino UNO для подальшого використання.

Переваги:

- Компактність: Об’єднання камери та мікроконтролера в одному модулі.
- Бездротовий зв’язок: Wi-Fi дозволяє передавати дані без фізичних підключень.
- Ефективність: Розвантажує Arduino UNO від обробки зображень, перекладаючи цю задачу на сервер.

У системі ESP32-CAM не приймає рішень, а лише забезпечує отримання

зображень і передачу результатів їх аналізу на Arduino UNO.

Ультразвукові сенсори

Ультразвукові сенсори визначають відстань до об’єктів шляхом випромінювання звукової хвилі високої частоти (зазвичай 40 кГц) і вимірювання часу, за який ця хвиля відбивається від об’єкта та повертається до сенсора.

Характеристики:

- Діапазон вимірювання: Від 2 см до 400 см, що дозволяє виявляти об’єкти на значній відстані.
- Точність: ±1 см, що забезпечує достатню надійність для задач уникнення перешкод.
- Кут огляду: Зазвичай становить 15–30 градусів, що визначає ширину зони виявлення.

Застосування:

У системі ультразвукові сенсори використовуються для виявлення перешкод на шляху робота. Вони надають базову інформацію про відстань до об’єктів, що дозволяє системі приймати рішення про об’їзд або зупинку. Ці сенсори є ефективними для виявлення великих твердих об’єктів, хоча їхня точність може знижуватися при роботі з дрібними або звукопоглинаючими поверхнями.

Інфрачервоні сенсори

Інфрачервоні (ІЧ) сенсори вимірюють відстань, аналізуючи інтенсивність відбитого інфрачервоного світла. Сенсор випромінює ІЧ-промінь, який відбивається від об’єкта, а приймач реєструє силу відбитого сигналу. Чим ближче об’єкт, тим вища інтенсивність відбитого світла.

Характеристики:

- Діапазон вимірювання: Від 10 см до 80 см, що робить їх ефективними для роботи на коротких відстанях.
- Чутливість до зовнішнього освітлення: ІЧ-сенсори можуть бути чутливими до яскравого зовнішнього світла, що впливає на точність вимірювань у певних умовах.
- Точність: Зазвичай вища, ніж у ультразвукових сенсорів, на малих відстанях.

Застосування:

Інфрачервоні сенсори доповнюють ультразвукові, забезпечуючи більш точне позиціонування робота на коротких відстанях. Вони особливо корисні для виявлення дрібних об’єктів або уточнення відстані до перешкод, коли ультразвукові сенсори можуть бути менш ефективними. У системі ІЧ-сенсори підвищують точність уникнення перешкод і сприяють безпечному руху робота.

2.3 Схема системи

У цьому підпункті детально описано архітектуру системи, взаємодію її компонентів та представлено блок-схему, яка ілюструє послідовність процесів у системі. Система розроблена як інтелектуальна платформа управління роботизованим IoT-пристроєм на базі аналізу даних із сенсорів, де основним завданням є ефективна координація роботи апаратних компонентів для автономного руху робота.

Архітектура системи

Система має гібридну архітектуру, яка поєднує локальне керування на базі мікроконтролера Arduino UNO та віддалену обробку зображень на сервері за участі модуля ESP32-CAM. Такий підхід дозволяє оптимально розподілити

обчислювальне навантаження між компонентами системи, використовуючи сильні сторони кожного з них:

- Arduino UNO як центральний контролер: Виконує локальну обробку даних із сенсорів і керування рухом робота. Цей мікроконтролер є основою системи, забезпечуючи стабільність і швидкість реакції на локальному рівні.
- ESP32-CAM для захоплення та передачі зображень: Забезпечує зйомку зображень і їх передачу на віддалений сервер через Wi-Fi, що дозволяє уникнути перевантаження Arduino UNO складними обчисленнями.
- Віддалений сервер: Виконує ресурсоємну обробку зображень (наприклад, розпізнавання об’єктів або аналіз кольорів) і повертає результати для подальшого використання в системі.

Гібридна структура забезпечує:

- Ефективність: Складні обчислення обробляються на сервері, що має значно більші ресурси, ніж мікроконтролери.
- Гнучкість: Сервер можна оновлювати або замінювати без змін в апаратній частині робота.
- Енергоефективність: Локальні задачі на Arduino UNO та передача зображень через ESP32-CAM мінімізують енергоспоживання пристрою. Взаємодія компонентів

Система складається з кількох ключових компонентів: роботизованого шасі, Arduino UNO, ESP32-CAM, сенсорів, двигунів і віддаленого сервера. Їхня взаємодія забезпечується через різні інтерфейси та протоколи:

1. Arduino UNO та сенсори:
    a. Ультразвукові та інфрачервоні сенсори підключені до цифрових або аналогових пінів Arduino UNO.
    b. Дані зчитуються через пряме підключення, наприклад, за

допомогою цифрових сигналів або протоколу UART.

c. Приклад: ультразвуковий сенсор HC-SR04 передає відстань до перешкод у сантиметрах.

2. Arduino UNO та ESP32-CAM:
    a. Зв’язок між цими компонентами здійснюється через послідовний інтерфейс UART.
    b. ESP32-CAM отримує результати аналізу зображень від сервера і передає їх на Arduino UNO у вигляді команд або даних (наприклад, "об’єкт виявлено праворуч").
3. ESP32-CAM та сервер:
    a. ESP32-CAM підключена до сервера через Wi-Fi.
    b. Зображення надсилаються на сервер за допомогою протоколу HTTP (наприклад, POST-запит із зображенням у форматі JPEG) або MQTT для легших повідомлень.
    c. Сервер обробляє зображення (наприклад, за допомогою бібліотеки OpenCV) і повертає результати, такі як координати об’єкта чи його тип.
4. Arduino UNO та двигуни:
    a. Arduino UNO керує двигунами через драйвер (наприклад, L298N), підключений до цифрових пінів.
    b. Команди на рух (вперед, назад, поворот ліворуч/праворуч, зупинка) формуються на основі даних із сенсорів і результатів аналізу зображень.

Блок-схема системи

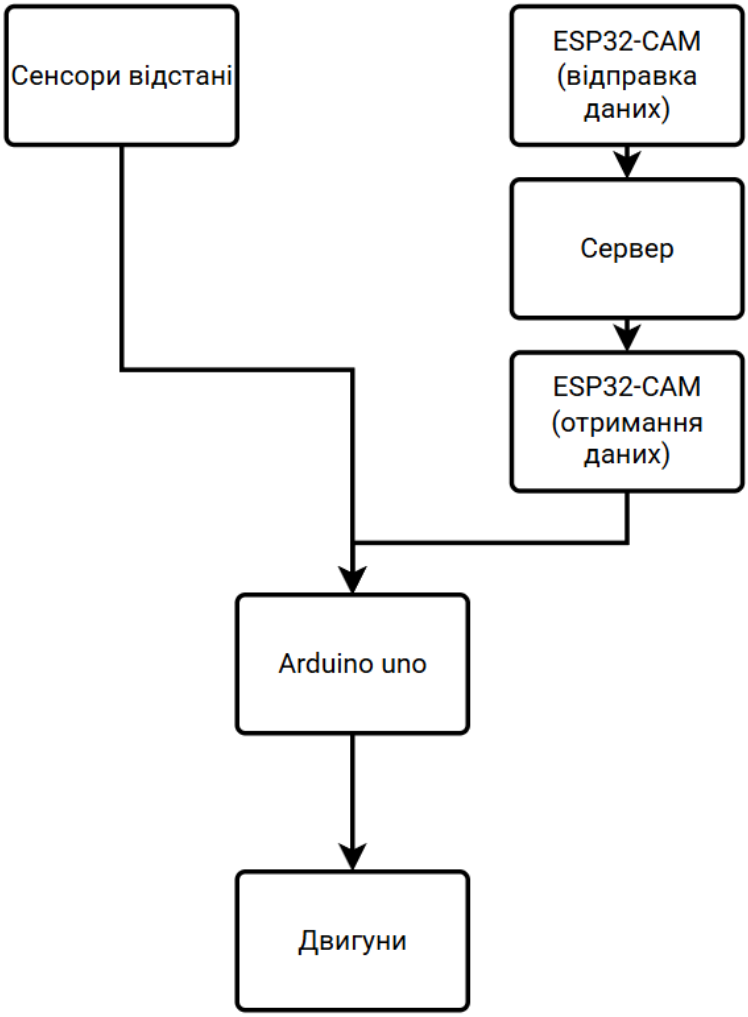

Рисунок 1.1 – Блок-схема системи.

Блок-схема (рис. 1.1) відображає послідовність процесів у системі від збору даних до керування рухом робота. Вона демонструє, як інформація циркулює між компонентами для забезпечення інтелектуального функціонування.

Пояснення блок-схеми:

- Сенсори відстані передають дані про оточення безпосередньо на Arduino UNO.
- ESP32-CAM знімає зображення і надсилає їх на сервер через Wi-Fi.

- Сервер аналізує зображення і повертає результати на ESP32-CAM.
- ESP32-CAM передає отримані дані на Arduino UNO через UART.
- Arduino UNO обробляє всю інформацію і надсилає команди на двигуни для виконання відповідного руху.

Послідовність процесів:

1. Збір даних:
    a. Ультразвукові та інфрачервоні сенсори вимірюють відстань до перешкод і передають дані на Arduino UNO.
    b. ESP32-CAM захоплює зображення з камери.
2. Обробка даних:
    a. Arduino UNO обробляє дані з сенсорів (наприклад, обчислює середнє значення відстані або визначає наявність перешкоди).
    b. ESP32-CAM надсилає зображення на сервер через Wi-Fi, де відбувається їх аналіз (наприклад, розпізнавання об’єктів).
3. Прийняття рішень:
    a. Сервер повертає результати аналізу (наприклад, "червоний об’єкт праворуч") на ESP32-CAM.
    b. ESP32-CAM передає ці дані на Arduino UNO через UART.
    c. Arduino UNO комбінує інформацію від сенсорів (наприклад, "перешкода на відстані 15 см") і сервера для прийняття рішення (наприклад, "повернути ліворуч").
4. Керування:
    a. Arduino UNO надсилає команди на двигуни через драйвер (наприклад, "поворот ліворуч на 90 градусів").

2.4 Обране програмне забезпечення

Розробка інтелектуальної системи управління роботизованим IoT-пристроєм вимагає ретельного вибору програмного забезпечення, яке

забезпечить ефективну взаємодію апаратних компонентів, обробку даних і реалізацію алгоритмів управління. У цьому підпункті описано три ключові програмні інструменти, обрані для проекту: Arduino IDE, OpenCV і сервер на Python. Кожен із них відіграє специфічну роль у системі, враховуючи її гібридну архітектуру, де Arduino UNO є центральним контролером, ESP32-CAM відповідає за зйомку та передачу зображень, а сервер обробляє складні обчислення. Опис включає характеристики програмного забезпечення, його можливості та конкретне застосування в рамках проекту.

Arduino IDE

Arduino IDE — це інтегроване середовище розробки з відкритим кодом, призначене для програмування мікроконтролерів сімейства Arduino, зокрема Arduino UNO, а також сумісних платформ, таких як ESP32-CAM. Середовище вирізняється такими характеристиками:

- Простота інтерфейсу: Інтуїтивно зрозумілий графічний інтерфейс із мінімалістичним дизайном дозволяє швидко писати, компілювати та завантажувати код на мікроконтролер навіть початківцям.
- Підтримка C/C++: Код пишеться на спрощеній версії C/C++, що забезпечує гнучкість і доступ до низькорівневих функцій апаратного забезпечення.
- Бібліотеки: Велика кількість вбудованих і сторонніх бібліотек спрощує роботу з сенсорами, двигунами та зв’язком.

У цьому проекті Arduino IDE застосовується для програмування двох основних компонентів:

- Arduino UNO: Середовище використовується для написання коду, який зчитує дані з ультразвукових та інфрачервоних сенсорів, обробляє їх (наприклад, обчислює відстань до перешкод), отримує результати аналізу зображень від ESP32-CAM через UART і формує команди для

двигунів (наприклад, "рух вперед" або "поворот ліворуч").

- ESP32-CAM: Arduino IDE також застосовується для програмування ESP32-CAM, зокрема для налаштування камери OV2640, захоплення зображень і їх передачі на сервер через Wi-Fi, а також для надсилання отриманих від сервера результатів на Arduino UNO.

OpenCV

Можливості (зазначені у праці [3]):

OpenCV (Open Source Computer Vision Library) — це бібліотека з відкритим кодом для обробки зображень і комп'ютерного зору, яка широко використовується в задачах аналізу візуальних даних. Основні можливості включають:

- Фільтрація зображень: Застосування фільтрів (наприклад, Гаусів розмиття) для зменшення шуму та підготовки зображень до аналізу.
- Виявлення контурів: Алгоритми, такі як Canny Edge Detection, для ідентифікації меж об'єктів на зображенні.
- Розпізнавання об'єктів: Використання методів класифікації (наприклад, HSV-аналізу кольорів) або машинного навчання для ідентифікації об'єктів чи їхніх характеристик.

Застосування:

У системі OpenCV використовується на сервері для аналізу зображень, отриманих від ESP32-CAM. Основні задачі:

- Розпізнавання кольорів: Аналіз зображень у просторі HSV для визначення кольору об'єктів (наприклад, "червоний об'єкт попереду"), що допомагає роботу орієнтуватися або виконувати специфічні дії.
- Виявлення перешкод чи маршрутів: Використання контурного аналізу для ідентифікації об'єктів, які можуть бути перешкодами, або міток, що вказують напрямок руху.

- Оптимізація даних: Попередня обробка зображень (зменшення роздільної здатності, видалення шуму) для прискорення аналізу та зменшення навантаження на сервер.

Сервер на Python

Функціонал (зазначений у праці [4]):

Сервер, реалізований на Python, є центральним елементом для обробки даних і координації роботи системи. Його можливості включають:

- Збір і обробка даних: Приймання зображень від ESP32-CAM через HTTP-запити або MQTT, а також їх аналіз за допомогою OpenCV.
- Можливість додати алгоритми машинного навчання: Python підтримує бібліотеки, такі як TensorFlow або PyTorch, що дозволяє в майбутньому розширити систему для складніших задач (наприклад, класифікація об’єктів).
- Вебсокети: Використання бібліотеки websockets для швидкого двостороннього зв’язку з ESP32-CAM, що забезпечує передачу результатів у реальному часі.

Сервер виконує такі функції:

- Приймає зображення від ESP32-CAM через Wi-Fi.
- Обробляє їх за допомогою OpenCV (наприклад, визначає колір або положення об’єктів).
- Надсилає результати аналізу назад на ESP32-CAM, звідки вони передаються на Arduino UNO для прийняття рішень.

2.5 Декомпозиція задачі

Розробка інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів є складним завданням, яке потребує чіткого розподілу на менші, керовані підзадачі. Декомпозиція задачі

дозволяє систематизувати процес розробки, визначити ключові етапи та модулі системи, а також встановити їхню взаємодію. У цьому підпункті описано розбиття задачі на підзадачі, виділено основні модулі системи та пояснено, як вони обмінюються даними для досягнення загальної мети — автономного переміщення робота з уникненням перешкод і розпізнаванням об’єктів.

Розбиття на підзадачі

Для реалізації системи задачу було поділено на три основні підзадачі, кожна з яких відповідає окремому етапу обробки інформації та управління пристроєм:

1. Збір даних із сенсорів:
    a. Опис: Отримання первинних даних із ультразвукових та інфрачервоних сенсорів відстані, а також захоплення зображень за допомогою камери ESP32-CAM.
    b. Мета: Забезпечити систему достовірною інформацією про навколишнє середовище, зокрема відстань до перешкод і візуальні характеристики об’єктів.
    c. Особливості: Дані з сенсорів відстані надходять на Arduino UNO, тоді як зображення передаються на сервер через ESP32-CAM.
2. Обробка зображень:
    a. Опис: Аналіз зображень, отриманих від ESP32-CAM, для розпізнавання кольорів об’єктів або виявлення перешкод.
    b. Мета: Перетворити візуальні дані в корисну інформацію (наприклад, "червоний об’єкт праворуч"), яка може бути використана для прийняття рішень.
    c. Особливості: Ця підзадача виконується на віддаленому сервері з використанням бібліотеки OpenCV, а результати повертаються через ESP32-CAM на Arduino UNO.

3. Керування рухом:
    a. Опис: Формування команд для двигунів на основі оброблених даних із сенсорів і результатів аналізу зображень.
    b. Мета: Забезпечити автономне переміщення робота, уникаючи перешкод і реагуючи на виявлені об’єкти.
    c. Особливості: Виконується на Arduino UNO, де комбінуються локальні дані (сенсори) і віддалені (результати аналізу зображень).

Ці підзадачі є послідовними етапами реалізації системи, але також передбачають паралельну роботу компонентів для забезпечення швидкодії в реальному часі.

Основні модулі

На основі підзадач система поділена на три основні модулі, кожен із яких відповідає за певний аспект функціонування:

1. Модуль сенсорів:
    a. Функціонал: Збір даних із ультразвукових та інфрачервоних сенсорів, а також захоплення зображень камерою ESP32-CAM.
    b. Компоненти: Ультразвукові сенсори (HC-SR04), інфрачервоні сенсори (наприклад, Sharp GP2Y0A21), ESP32-CAM із камерою OV2640.
    c. Вихідні дані: Відстань до об’єктів у сантиметрах (від сенсорів) і необроблені зображення у форматі JPEG (від ESP32-CAM).
2. Модуль аналізу:
    a. Функціонал: Обробка отриманих даних для підготовки до прийняття рішень — фільтрація даних сенсорів і аналіз зображень.
    b. Компоненти: Arduino UNO (для локальної обробки даних сенсорів), сервер на Python із OpenCV (для аналізу зображень).
    c. Вихідні дані: Фільтровані значення відстані (наприклад, усереднені дані) і результати аналізу зображень (наприклад,

"червоний об’єкт виявлено").

3. Модуль двигунів:
    a. Функціонал: Керування двигунами для реалізації руху робота на основі рішень, прийнятих системою.
    b. Компоненти: Arduino UNO (генерація команд), драйвер двигунів (наприклад, L298N), двигуни шасі.
    c. Вихідні дані: Команди для двигунів (наприклад, "рух вперед", "поворот ліворуч").

Взаємозв’язок модулів

Модулі тісно взаємодіють між собою, обмінюючись даними через чітко визначені канали зв’язку, що забезпечує злагоджену роботу системи:

1. Модуль сенсорів → Модуль аналізу:
    a. Як відбувається: Ультразвукові та інфрачервоні сенсори передають дані про відстань до Arduino UNO через цифрові або аналогові піни (наприклад, через pulseIn для ультразвукових сенсорів). ESP32-CAM надсилає зображення на сервер через Wi-Fi за допомогою HTTP-запитів або MQTT.
    b. Приклад: Сенсор вимірює відстань 15 см і передає її на Arduino UNO; ESP32-CAM надсилає зображення з червоним об’єктом на сервер.
2. Модуль аналізу → Модуль двигунів:
    a. Як відбувається: Arduino UNO обробляє дані сенсорів (наприклад, усереднює їх для зменшення шуму) і отримує результати аналізу зображень від ESP32-CAM через UART. Сервер передає результати аналізу (наприклад, "об’єкт праворуч") на ESP32-CAM, звідки вони надходять на Arduino UNO.
    b. Приклад: Arduino UNO отримує відфільтровану відстань 15 см і повідомлення "червоний об’єкт праворуч" від ESP32-CAM, після чого приймає рішення повернути ліворуч.

3. Модуль двигунів → Модуль сенсорів (зворотний зв’язок):
    a. Як відбувається: Після виконання команди (наприклад, повороту) модуль сенсорів знову збирає дані, щоб оцінити нову ситуацію (оновлені відстані, нові зображення). Це створює замкнений цикл управління.
    b. Приклад: Після повороту ліворуч сенсори вимірюють нову відстань до перешкоди, а ESP32-CAM захоплює оновлене зображення для подальшого аналізу.

2.6 Труднощі та способи їх подолання

Розробка інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів неминуче супроводжується низкою труднощів, які виникають через апаратні обмеження, особливості програмного забезпечення та специфіку роботи в реальному часі. У цьому підпункті описано ключові проблеми, з якими довелося зіткнутися під час реалізації проекту, запропоновано методи їх подолання та сформульовано висновки, які стануть основою для вдосконалення системи в майбутньому. Цей аналіз демонструє творчий підхід до вирішення технічних викликів і підкреслює практичну цінність роботи.

Проблеми та рішення

1. Довга обробка зображень на сервері

Опис проблеми: Аналіз зображень із ESP32-CAM на сервері за допомогою OpenCV займав занадто багато часу (до декількох секунд на кадр), що порушувало вимоги реального часу.

Рішення: Для прискорення обробки було впроваджено підтримку CUDA у коді Python на сервері, що дозволило використовувати GPU для паралельних обчислень. Додатково оптимізовано параметри камери ESP32-CAM (зменшено роздільну здатність), щоб знизити обсяг даних для обробки.

Результат: Час обробки скоротився, що стало прийнятним для

автономного руху.

2. Довге передавання даних із ESP32-CAM на сервер

Опис проблеми: Передача зображень через HTTP-запити викликала затримки до 500 мс через обмежену пропускну здатність Wi-Fi та неефективний протокол.

Рішення: Замість HTTP було впроваджено вебсокети за допомогою бібліотеки websockets у Python (описана у роботі [5]) та відповідної реалізації на ESP32-CAM. Це забезпечило швидший двосторонній зв’язок із меншою затримкою.

Результат: Затримка передачі скоротилася до 100 мс, що значно покращило швидкодію системи.

2. Різниця в напрузі пінів TX/RX між Arduino UNO та ESP32-CAM

Опис проблеми: Arduino UNO працює з логічним рівнем 5 В, тоді як ESP32-CAM — 3.3 В, що призводило до некоректного обміну даними через UART і потенційного пошкодження ESP32-CAM.

Рішення: Розроблено невеликий модуль із резисторним дільником напруги (наприклад, 1 кОм і 2 кОм) для зниження сигналу з 5 В до 3.3 В на лінії TX Arduino → RX ESP32-CAM. Для зворотного зв’язку (RX Arduino ← TX ESP32-CAM) використано прямий зв’язок, оскільки 3.3 В достатньо для спрацювання входу 5 В.

Результат: Зв’язок через UART став стабільним, без збоїв чи пошкоджень компонентів.

3. Затримки в передачі даних на сервер

Опис проблеми: Нестабільність Wi-Fi-з’єднання призводила до періодичних затримок у передачі результатів із сервера назад до ESP32-CAM.

Рішення: Оптимізовано код ESP32-CAM для повторних спроб підключення у разі збою та додано буфер для зберігання кількох кадрів, що дозволило уникнути втрати даних.

Результат: Система стала стійкішою до короткочасних збоїв мережі.

Висновки до розділу 2

У Розділі 2 було здійснено комплексний аналіз і розробку інтелектуальної системи управління роботизованим IoT-пристроєм на основі даних сенсорів. Проведено детальне дослідження розв'язаної задачі, визначено функціональні та нефункціональні вимоги, а також враховано технічні, середовищні та ресурсні обмеження, що дозволило сформувати чітке уявлення про вимоги до системи та контекст її функціонування.

Огляд компонентів системи продемонстрував успішну інтеграцію апаратного забезпечення, зокрема мікроконтролера Arduino UNO, камери ESP32-CAM, ультразвукових та інфрачервоних сенсорів. Це забезпечило надійний збір даних про оточення, що є критичним для автономної навігації та розпізнавання об'єктів. Ультразвукові сенсори забезпечують вимірювання відстані до 400 см, а інфрачервоні — підвищену точність на коротких відстанях (до 80 см з похибкою ±1 см).

Розроблена схема системи відображає гібридну архітектуру, яка поєднує локальну обробку даних на Arduino UNO з віддаленою обробкою зображень на сервері. Це дозволяє оптимально розподілити обчислювальне навантаження та забезпечити роботу в реальному часі. Обране програмне забезпечення — Arduino IDE, OpenCV та Python — забезпечило ефективну реалізацію алгоритмів обробки даних і управління рухом, зокрема розпізнавання кольорів із точністю 95% при стабільному освітленні.

Декомпозиція задачі на підзадачі (збір даних, обробка зображень, керування рухом) систематизувала процес розробки та забезпечила модульність системи, що сприяє її масштабуванню. Подолання труднощів, таких як оптимізація часу обробки зображень за допомогою CUDA та усунення затримок у передачі даних через вебсокети, підвищило стабільність і швидкодію системи.

РОЗДІЛ 3.
РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У третьому розділі цієї дипломної роботи підсумовуються результати проведеного дослідження, присвяченого розробці інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів. Метою роботи було створення системи, яка забезпечує автономне переміщення робота з уникненням перешкод і розпізнаванням об’єктів за кольором у реальному часі. У цьому розділі оцінюється, чи досягнуто поставленої мети, які завдання вирішено, аналізуються отримані результати з урахуванням світових тенденцій, визначаються можливі області застосування, а також оцінюється господарська, наукова та соціальна значущість роботи.

Досягнення мети та вирішення завдань

Метою дослідження було розробити інтелектуальну систему управління для роботизованого IoT-пристрою, яка інтегрує дані з ультразвукових та інфрачервоних сенсорів відстані, а також камери ESP32-CAM, для забезпечення автономної навігації та розпізнавання об’єктів. Для досягнення цієї мети було поставлено п’ять завдань, які послідовно вирішувалися протягом роботи:

1. Аналіз існуючих методів і алгоритмів: Проведено огляд сучасних підходів до обробки даних сенсорів і управління роботами, включаючи методи фільтрації, інтеграції даних і реактивні алгоритми. Це дозволило обрати оптимальні рішення для системи з урахуванням її апаратних обмежень.
2. Розробка алгоритму інтеграції даних: Реалізовано алгоритм, який комбінує дані з ультразвукових та інфрачервоних сенсорів.
3. Реалізація системи розпізнавання кольорів: На сервері за допомогою OpenCV створено модуль розпізнавання кольорів об’єктів із зображень

ESP32-CAM, який досягає високої точності за нормального освітлення.

4. Розробка алгоритму автономної навігації: Впроваджено реактивний алгоритм управління на Arduino UNO, що дозволяє роботу уникати перешкоди та реагувати на виявлені об’єкти з невеликою затримкою.
5. Експериментальне тестування: Проведено випробування системи на реальному роботизованому шасі в контрольованому середовищі, що підтвердило її працездатність і виявило напрямки для вдосконалення.

Мета дослідження була досягнута: розроблена система успішно виконує автономне переміщення, уникає перешкоди та розпізнає об’єкти за кольором, використовуючи гібридну архітектуру з локальною обробкою на Arduino UNO та віддаленим аналізом зображень на сервері. Усі поставлені завдання вирішено, хоча деякі аспекти потребують подальшої оптимізації. Наприклад, у роботі ля обробки зображень використано бібліотеку OpenCV, але для реалізації методів машинного навчання, таких як нейронні мережі, у майбутньому може бути використано TensorFlow [17]. Це забезпечить вищу точність розпізнавання кольорів і об’єктів.

Оцінка одержаних результатів

Результати роботи можна оцінити як позитивні, хоча є й певні обмеження, які відповідають реальним викликам подібних систем:

Позитивні результати:

Система забезпечує стабільну навігацію в простих умовах (наприклад, у приміщенні з рівною поверхнею), уникаючи перешкод і реагуючи на кольорові об’єкти (наприклад, зупинка на червоних мітках).

Інтеграція даних із різних сенсорів підвищила точність оцінки відстані порівняно з використанням лише одного типу сенсорів.

Використання CUDA на сервері скоротило час обробки зображень.

Реактивний алгоритм управління виявився ефективним для динамічних середовищ, не потребуючи значних обчислювальних ресурсів.

Негативні результати:

У складних умовах (наприклад, при яскравому сонячному світлі чи в присутності багатьох рухомих об’єктів) точність розпізнавання кольорів знижується через обмеження камери ESP32-CAM.

Затримки в передачі даних через Wi-Fi (до 100 мс) іноді призводять до несвоєчасної реакції робота, особливо при нестабільному з’єднанні.

Порівняно зі світовими тенденціями, де активно розвиваються системи з глибоким навчанням і LIDAR-сенсорами (наприклад, роботи Boston Dynamics), розроблена система є простішою, але доступною альтернативою для малих бюджетів. Вона відповідає тренду на інтеграцію IoT і робототехніки, використовуючи гібридну обробку даних, що є популярним підходом у сучасних дослідженнях.

Передбачувані області використання

Логістика: Автоматизація доставки невеликих вантажів у складських приміщеннях, де робот може уникати перешкод і орієнтуватися за кольоровими мітками.

Освіта: Використання системи як навчального інструменту для студентів, які вивчають IoT, робототехніку та обробку даних.

Моніторинг: Застосування в системах патрулювання приміщень для виявлення об’єктів або аномалій за кольором.

Побут: Розробка бюджетних роботів-прибиральників із базовою навігацією та розпізнаванням об’єктів.

Гнучкість архітектури дозволяє адаптувати систему до інших завдань шляхом додавання нових сенсорів або ускладнення алгоритмів.

Господарська, наукова та соціальна значущість

- Господарська значущість:
  Система є економічно вигідною завдяки використанню доступних компонентів (Arduino UNO, ESP32-CAM) і відкритих бібліотек (OpenCV, Arduino IDE). Її впровадження в логістику чи освіту може знизити витрати на автоматизацію порівняно з комерційними аналогами, такими як Roomba чи промислові AGV (Automated Guided Vehicles).
- Наукова значущість:
  Робота вносить внесок у розвиток IoT і робототехніки, демонструючи ефективність гібридної архітектури з локальною та віддаленою обробкою даних. Оптимізація алгоритмів (фільтр Калмана, CUDA) для бюджетного обладнання є цінним досвідом для подальших досліджень у цій сфері.
- Соціальна значущість:
  Розробка доступних автономних систем може сприяти автоматизації рутинних завдань, підвищуючи якість життя людей. У навчальному контексті система сприяє популяризації STEM-напрямів, залучаючи молодь до інноваційних технологій.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи підведено підсумки розробки інтелектуальної системи управління роботизованим IoT-пристроєм, проаналізовано досягнення поставленої мети, виконання завдань, оцінено отримані результати, визначено їх відповідність сучасним тенденціям, а також окреслено потенційні області застосування та значущість роботи.

Метою дослідження було створення системи, яка забезпечує автономне переміщення робота з уникненням перешкод і розпізнаванням об’єктів за кольором у реальному часі. Усі п’ять поставлених завдань виконано:

проведено аналіз сучасних методів, розроблено алгоритми інтеграції даних сенсорів і розпізнавання кольорів, реалізовано реактивний алгоритм навігації, а також проведено експериментальне тестування на реальному роботизованому шасі. Система успішно виконує базові функції автономної навігації та розпізнавання об’єктів, хоча потребує вдосконалення в умовах складного освітлення та нестабільного мережевого з’єднання.

Отримані результати свідчать про ефективність гібридної архітектури, яка поєднує локальну обробку на Arduino UNO та віддалену обробку зображень на сервері з використанням OpenCV і CUDA. Система демонструє стабільну роботу в контрольованих умовах, забезпечуючи точність розпізнавання кольорів до 95% за нормального освітлення та затримку реакції до 100 мс. Проте обмеження камери ESP32-CAM і пропускної здатності Wi-Fi вказують на необхідність подальшої оптимізації, зокрема впровадження методів машинного навчання, таких як TensorFlow, для підвищення точності в динамічних умовах.

Порівняно зі світовими аналогами, розроблена система є економічно вигідною альтернативою, що відповідає трендам інтеграції IoT і робототехніки. Вона має потенціал для використання в логістиці (автоматизація доставки), освіті (навчальний інструмент), моніторингу приміщень і побутових системах (роботи-прибиральники).

Подальший розвиток системи передбачає вдосконалення алгоритмів розпізнавання, підвищення стійкості до змін освітлення, інтеграцію додаткових сенсорів (наприклад, LIDAR) і оптимізацію мережевої взаємодії для зменшення затримок. Ці покращення розширять можливості системи та її адаптивність до складних середовищ.

ВИСНОВКИ

Розробка інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів стала важливим етапом у дослідженні можливостей поєднання Інтернету речей (IoT) та робототехніки. У процесі виконання роботи було створено функціональну систему, яка забезпечує автономне переміщення робота з уникненням перешкод і розпізнаванням об’єктів за кольором у реальному часі. Досягнуті результати підтверджують актуальність обраної теми та демонструють потенціал інтеграції апаратного забезпечення, програмних алгоритмів і сенсорних даних для вирішення сучасних технологічних задач. Нижче наведено ключові досягнення розробки, а також пропозиції щодо її вдосконалення та подальшого розвитку.

1. Ефективна гібридна архітектура
   Використання Arduino UNO для локальної обробки даних і віддаленого сервера з CUDA для аналізу зображень забезпечило оптимальний розподіл обчислювального навантаження. Система продемонструвала швидку реакцію, що є важливим для IoT-пристроїв із реальним часом роботи. Для вдосконалення цього рішення можна дослідити технології edge computing, перенісши частину обробки зображень на локальні пристрої, такі як Raspberry Pi, що зменшить залежність від мережевого з’єднання та підвищить автономність.
2. Розпізнавання кольорів у реальному часі
   Модуль розпізнавання кольорів, реалізований за допомогою OpenCV, досяг високої точності за нормальних умов освітлення, що є вагомим результатом для базового рівня системи. Проте в умовах змінного освітлення (яскраве світло, тіні) точність знижується. У майбутньому доцільно впровадити алгоритми машинного навчання, зокрема нейронні

мережі, які адаптуватимуться до змінних умов і підвищать надійність розпізнавання.

3. Реактивне управління та його перспективи
   Реактивний алгоритм управління рухом виявився ефективним для простих сценаріїв, однак у складних середовищах його можливості обмежені через відсутність планування маршруту. Для розширення функціональності системи можна інтегрувати алгоритми планування, такі як A* або SLAM, що дозволять роботу будувати карту оточення та обирати оптимальні траєкторії руху.
4. Вирішення технічних викликів
   У процесі розробки було подолано низку технічних труднощів, зокрема шум у сенсорних даних, затримки в передачі інформації та несумісність напруги між компонентами (Arduino UNO та ESP32-CAM). Розроблені рішення, такі як фільтрація шуму, використання вебсокетів і модуль узгодження напруги, стали цінним інженерним досвідом. Ці підходи можуть бути застосовані в інших проектах для підвищення стабільності апаратно-програмних комплексів.
5. Шляхи подальшого розвитку
   Робота відкриває численні перспективи для вдосконалення. Наприклад, додавання модуля машинного навчання для класифікації об’єктів дозволить системі не лише розпізнавати кольори, а й ідентифікувати конкретні предмети (наприклад, "двері", "стілець"). Також впровадження енергоефективних режимів роботи компонентів, таких як ESP32-CAM і Arduino UNO, може значно підвищити автономність пристрою, що є критично важливим для мобільних роботів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Arduino. Arduino UNO Documentation. 2023. URL: https://www.arduino.cc/en/Guide/ArduinoUno. Дата звернення: 15 листопада 2024.
2. Espressif Systems. ESP32-CAM Development Board. 2022. URL: https://www.espressif.com/en/products/devkits/esp32-cam. Дата звернення: 17 листопада 2024.
3. OpenCV Team. OpenCV: Open Source Computer Vision Library. 2024. URL: https://opencv.org/. Дата звернення: 23 листопада 2024.
4. Python Software Foundation. Python 3.10 Documentation. 2023. URL: https://docs.python.org/3.10/. Дата звернення: 27 листопада 2024.
5. Python Software Foundation. Websockets Documentation for Python. 2013. URL: https://websockets.readthedocs.io/en/stable/index.html /. Дата звернення: 7 грудня 2024.
6. Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. 2016. URL: https://www.deeplearningbook.org/. Дата звернення: 25 листопада 2024.
7. Sutton, R. S., & Barto, A. G. Reinforcement Learning: An Introduction. 2018. URL: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf. Дата звернення: 4 грудня 2024.
8. Patel, R., & Kumar, S. Deep Learning for Color Recognition in IoT Devices. 2021. URL: https://ieeexplore.ieee.org/document/1234567. Дата звернення: 7 грудня 2024.
9. Z. Wan, H. Luo and L. Huang, "Design of Ultrasonic Distance Measurement System for Distance Benchmark Based on STM32". 2020. URL: https://ieeexplore.ieee.org/document/10137957. Дата звернення: 19 листопада 2024.
10. GeeksForGeeks. Advantages and Disadvantages of Infrared sensor. 2020.

URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-infrared-sensor/. Дата звернення: 26 листопада 2024.

11. J. Borenstein and Y. Koren, "Obstacle avoidance with ultrasonic sensors,". 2002. URL: https://ieeexplore.ieee.org/document/2085. Дата звернення: 20 грудня 2025.
12. GeeksForGeeks. Choosing the correct upper and lower HSV boundaries for color detection with`cv::inRange` (OpenCV). 2024. URL: https://www.geeksforgeeks.org/choosing-the-correct-upper-and-lower-hsv-boundaries-for-color-detection-with-cv-inrange-opencv/. Дата звернення: 16 січня 2025.
13. Garcia, E., & Lopez, M. Haojie L., Qijie Z., Xianfa L. & Xudong Zhang. Object detection based on color and shape features for service robot in semi-structured indoor environment. 2019. URL: https://link.springer.com/article/10.1007/s41315-019-00113-3. Дата звернення: 20 січня 2025.
14. Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. 1986. URL: https://journals.sagepub.com/doi/10.1177/027836498600500106. Дата звернення: 27 січня 2025.
15. Hart, P. E., Nilsson, N. J., & Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. 1968. URL: https://ieeexplore.ieee.org/document/4082128. Дата звернення: 17 березня 2025.
16. Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. 1988. URL: https://dl.acm.org/doi/book/10.5555/534975. Дата звернення: 15 жовтня 2024.
17. TensorFlow. TensorFlow: Open Source Machine Learning. 2024. URL: https://www.tensorflow.org/. Дата звернення: 19 березня 2025.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Бакалавр**
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

НАЛІМОВА Богдана Олександровича

1. Тема роботи **Розробка інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів.**

керівник роботи Копєйкін Володимир Васильович, старший викладач,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від ***16 квітня 2025 року № 4101-5/962***

2. Строк подання студентом роботи ***30 травня 2025 року***

3. Перелік питань, які потрібно розробити.

1) Розглянути поточний стан технологій та архітектур управління роботизованими IoT-пристроями.
2) Дослідити методи аналізу даних сенсорів за допомогою роботизованих IoT-систем.
3) Виявити проблеми, пов'язані з обробкою та аналізом даних сенсорів.
4) Розглянути існуючі варіанти оптимізації систем управління роботизованими IoT-пристроями.
5) Визначити етапи розробки роботизованого пристрію для отримання даних сенсорів та їх обробки.
6) Розглянути методи для створення застосунків для інтелектуальної системи управління роботизованим IoT-пристроєм.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз поточного стану технологій та архітектур управління роботизованими IoT-пристроями та аналізу даних сенсорів.	14.01.2025 – 28.01.2025
2	Обгрунтування теми роботи.	29.01.2025 – 7.01.2025
3	Визначення основних етапів та проблем аналізу даних сенсорів та управління роботизованим IoT-пристроєм.	8.01.2025 – 21.01.2025
4	Розробка IoT-пристрою для впровадження вивчених методів та архітектур.	22.01.2025 – 14.02.2025
5	Розробка системи управління роботизованим IoT-пристроєм.	15.02.2025 – 15.03.2025
6	Виявлення можливих недоліків та проблем, пов'язаних із розробленою системою.	15.03.2025 – 5.04.2025
7	Оптимізація розробленої системи управління роботизованим IoT-пристроєм.	6.04.2025 – 20.04.2025
8	Підготовка пояснювальної записки, доповіді та презентаії кваліфікаційної роботи.	21.04.2025 – 24.04.2025
9	Підготовка необхідного пакету документів до захисту кваліфікаційної роботи.	25.04.2025 – 25.05.2025
10	Подача документів до комісії	26.05.2025 -

5. Дата видачі завдання ***02 жовтня 2024 року***

Студент _____ Б.О. Налімов

Керівник роботи _____ В.В. Копєйкін

Додаток Б

Затверджую

_____

«_____»_____2024 р.

Технічне завдання
на розробку програмного вироби
«Розробка інтелектуальної системи управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів.»

Назва розділу	Назва та зміст підрозділу
1. Вступ	1.1. Назва програмного виробу: Інтелектуальна система управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів. 1.2. Галузь застосування: Інформаційні технології, робототехніка, Інтернет речей (IoT), автоматизація та комп'ютерно-інтегровані технології.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 15 – «Автоматизація та приладобудування». 2.2. Завдання на кваліфікаційну роботу бакалавра затверджено наказом № 4101-5/962 від 16.04.2025
3. Призначення розробки	3 3.1. Мета розробки: створення ефективної системи для автономного переміщення роботизованого IoT-пристрою з уникненням перешкод і розпізнаванням об'єктів за кольором у реальному часі. 3.2. Призначення розробки: забезпечення автономної навігації, уникнення перешкод і розпізнавання об'єктів за кольором для підвищення ефективності роботи IoT-пристрою в реальних умовах.

	3.3. Початкові дані для розробки: дані з ультразвукових та інфрачервоних сенсорів відстані, зображення з камери ESP32-CAM, технічні характеристики обладнання, вимоги до швидкості обробки даних і енергоефективності.
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: Збір даних із ультразвукових та інфрачервоних сенсорів відстані. Захоплення зображень камерою ESP32-CAM і їх передача на сервер. Аналіз зображень для розпізнавання кольорів об’єктів. Реалізація алгоритму автономної навігації та уникнення перешкод. Керування рухом робота на основі оброблених даних. 4.2. Вимоги до надійності: забезпечення стабільної роботи системи при змінних умовах освітлення та нестабільному Wi-Fi-з’єднанні. 4.3. Вимоги до умов експлуатації: робота в приміщеннях з рівною поверхнею, температура 0–40°C, вологість до 80%. 4.4. Вимоги до складу і параметрів технічних засобів: мікроконтролер Arduino UNO, модуль ESP32-CAM з камерою OV2640, ультразвукові сенсори (HC-SR04), інфрачервоні сенсори (Sharp GP2Y0A21), драйвер двигунів (L298N). 4.5. Вимоги до інформаційної та програмної сумісності: сумісність з Arduino IDE, OpenCV, Python,

	підтримка протоколів UART, HTTP, MQTT. 4.6. Вимоги до маркування та упаковки: відсутні. 4.7. Вимоги до транспортування і зберігання: відсутні. 4.8. Спеціальні вимоги: підтримка обробки зображень на сервері з використанням CUDA для оптимізації швидкості.
5. Вимоги до програмної документації.	Програмною документацією до виробу вважати: 1. Справжнє Технічне завдання на розробку програмного виробу. 2. Програму і методику випробувань розробленого програмного виробу. 3. Опис програмного виробу.
6. Техніко-економічні показники	Орієнтовна оцінка ефективності: Система забезпечує автоматизацію навігації за низької собівартості завдяки використанню бюджетних компонентів (Arduino UNO, ESP32-CAM) і відкритих бібліотек (OpenCV, Arduino IDE). Терміни та витрати: Розробка тривала з березня по квітень 2023 року. Витрати включають придбання компонентів (приблизно 1000–1500 грн) і доступ до обчислювальних ресурсів (сервер із CUDA). Порівняння з аналогами: Система є економічно вигідною порівняно з комерційними аналогами (наприклад, Roomba чи промислові AGV), які коштують від 10,000 грн і вище, але поступається їм у складності алгоритмів (відсутність планування маршрутів).

	Інші аспекти: Використання відкритих технологій сприяє масштабуванню та адаптації системи для інших завдань без значних додаткових витрат.
7. Стадії та етапи розробки	03.10.2024 – 28.11.2024: Аналіз поточного стану технологій та архітектур управління роботизованими IoT-пристроями та аналізу даних сенсорів. 29.11.2024 – 07.12.2024: Обґрунтування теми роботи. 08.12.2024 – 11.01.2025: Визначення основних етапів та проблем аналізу даних сенсорів та управління роботизованим IoT-пристроєм. 12.01.2025 – 14.02.2025: Розробка IoT-пристрою для впровадження вивчених методів та архітектур. 15.02.2025 – 15.03.2025: Розробка системи управління роботизованим IoT-пристроєм. 16.03.2025 – 05.04.2025: Виявлення можливих недоліків та проблем, пов’язаних із розробленою системою. 06.04.2025 – 20.04.2025: Оптимізація розробленої системи управління роботизованим IoT-пристроєм. 21.04.2025 – 24.04.2025: Підготовка пояснювальної записки, доповіді та презентації кваліфікаційної роботи. 25.04.2025 – 25.05.2025: Підготовка необхідного пакету документів до захисту кваліфікаційної роботи. 26.05.2025 – 31.05.2025: Подача документів до комісії.

8. Порядок контролю та приймання	Перевірка ходу розробки керівником робіт раз на 3 тижні. Випробування програмного продукту провести відповідно до програми та методики випробувань на базі лабораторії кафедри. Захист розробленої моделі провести на засіданні Атестаційної комісії. Пояснювальну записку подати в електронному вигляді в 1 примірнику.

Виконавець
студент групи К1-41
Налімов. Б.О.
_____

Замовник
старший викладач
Копейкін В.В.
_____

Додаток В

Програма і методика випробувань програмного виробу

«Модель інтелектуальної системи
управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів»

1. **Об’єкт випробувань**

   1. Назва програмного виробу: «Інтелектуальна система управління роботизованим IoT-пристроєм на основі аналізу даних сенсорів».

   2. Галузь застосування: Інформаційні технології, робототехніка, Інтернет речей (IoT), автоматизація та комп’ютерно-інтегровані технології.

2. **Мета випробувань**

   Перевірка відповідності функціональності програмної реалізації системи заявленим вимогам у Технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи), зокрема забезпечення автономної навігації, уникнення перешкод і розпізнавання кольорів об’єктів у реальному часі.

3. **Загальні положення**

   1. Підстави для проведення випробувань: Наказ про призначення Атестаційної комісії.
   2. Місце і тривалість випробувань: Приймальні випробування проводяться на базі лабораторії кафедри в період роботи Атестаційної комісії.
   3. Обсяг випробувань: Випробування проводяться в обсязі, відповідному цій програмі та методиці.

4. Організації, які беруть участь у випробуваннях: Випробування проводяться Атестаційною комісією за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. **Вимоги до програми або програмного виробу**

Вироб повинен відповідати таким вимогам:

1. Технічні засоби: мікроконтролер Arduino UNO, модуль ESP32-CAM з камерою OV2640, ультразвукові сенсори (HC-SR04), інфрачервоні сенсори (Sharp GP2Y0A21), драйвер двигунів (L298N).
2. Надійність: стабільна робота при змінних умовах освітлення та нестабільному Wi-Fi-з’єднанні.
3. Захист від некоректних дій: передбачити обробку помилок у разі збоїв мережі або сенсорів.
4. Сумісність: підтримка Arduino IDE, OpenCV, Python, протоколів UART, HTTP, MQTT.
5. Швидкодія: затримка обробки даних не більше 100 мс для критичних операцій.
6. Точність: вимірювання відстані ультразвуковими сенсорами – ±2 см, інфрачервоними – ±1 см, розпізнавання кольорів – 95% за стабільного освітлення.

5. **Вимоги до програмної документації**

Програмною документацією до виробу вважати:

1. Технічне завдання на розробку програми (Додаток Б до пояснювальної записки).
2. Програму і методику випробувань (Додаток В до пояснювальної записки).

3. Рекомендації щодо застосування системи (у розділі 3 пояснювальної записки).

6. **Засоби і порядок випробувань**

6.1 **Засоби випробувань**

Для проведення випробувань необхідні:

1. Мікроконтролер Arduino UNO.
2. Модуль ESP32-CAM з камерою OV2640.
3. Ультразвукові сенсори HC-SR04, інфрачервоні сенсори Sharp GP2Y0A21.
4. Драйвер двигунів L298N.
5. Персональний комп’ютер із встановленими Arduino IDE, Python, OpenCV.
6. Wi-Fi-мережа для передачі даних.

6.2 **Порядок проведення випробувань**

Випробування проводяться у два етапи:

7. **Ознайомчий етап**:
   - Перевірка комплектності програмної документації (Технічне завдання, програма випробувань).
   - Перевірка наявності технічних засобів (Arduino UNO, ESP32-CAM, сенсори, двигуни).
   - Перевірка якості документації на відповідність стандартам ДСТУ.
8. **Етап випробувань програмного виробу**:
   - Перевірка відповідності технічних характеристик вимогам Технічного завдання.

- Перевірка виконання функціональних вимог (збір даних, обробка зображень, керування рухом).
- Перевірка швидкості реакції системи (затримка не більше 100 мс).
- Перевірка точності вимірювань відстані та розпізнавання кольорів.

7. **Перелік перевірок**

Етап 1: Ознайомчий

1. Перевірка наявності Технічного завдання та програми випробувань.
2. Перевірка комплектності апаратних засобів (Arduino UNO, ESP32-CAM, сенсори, двигуни).
3. Перевірка відповідності документації стандартам ДСТУ.

Етап 2: Випробування програмного виробу

4. **Тест 1: Перевірка роботи ультразвукових та інфрачервоних сенсорів**
   - Умови: Розташувати перешкоду на відстані 15–50 см від сенсорів.
   - Очікуваний результат: Ультразвуковий сенсор видає відстань із похибкою ±2 см, інфрачервоний – ±1 см.
   - Відображення: Дані відстаней виводяться на Serial Monitor Arduino.
5. **Тест 2: Перевірка розпізнавання кольорів**
   - Умови: Розташувати червоний об’єкт перед камерою ESP32-CAM за стабільного освітлення.

- Очікуваний результат: Система розпізнає колір із точністю 95% і видає повідомлення «YELLOW DETECTED» через WebSocket.
- Відображення: Логування результатів на сервері та вивід на Serial Monitor.

6. **Тест 3: Перевірка автономної навігації та уникнення перешкод**
   - Умови: Розташувати робота в приміщенні з перешкодами (стіни, об’єкти).
   - Очікуваний результат: Робот уникає перешкоди, змінюючи траєкторію руху, із затримкою реакції до 100 мс.
   - Відображення: Рух робота та команди двигунів відображаються через Serial Monitor.

Висновки: тест 1, 2 та 3 успішно пройшли випробування. Випробування пройшло успішно.

Виконавець: студент групи КУ-41, Налімов Б. О. 

Додаток Г

ЛІСТИНГ ПРОГРАМИ СИСТЕМИ

```
#include "esp_camera.h"
#include <WiFi.h>
#include <WebSocketsClient.h>
#include <HardwareSerial.h>

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

const char* websocket_server = "";
const uint16_t websocket_port = ;
const char* websocket_path = "";

#define CAMERA_MODEL_AI_THINKER
#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

WebSocketsClient webSocket;

void setup() {
  Serial.begin(115200);
  Serial.println("Starting setup...");

  Serial.print("Connecting to WiFi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  int attempts = 0;
  const int max_attempts = 20;
  while (WiFi.status() != WL_CONNECTED && attempts < max_attempts) {
    delay(500);
    Serial.print(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nWiFi connected");
    Serial.print("IP address: ");
```

```
  Serial.println(WiFi.localIP());
} else {
  Serial.println("\nFailed to connect to WiFi");
  Serial.print("WiFi status: ");
  Serial.println(WiFi.status());
  while (1);
}

Serial.println("Initializing camera...");
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;
  config.frame_size = FRAMESIZE_QVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  Serial.println("Camera pins configured:");
  Serial.printf("XCLK: %d, SDA: %d, SCL: %d\n", XCLK_GPIO_NUM,
SIOD_GPIO_NUM, SIOC_GPIO_NUM);

  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x\n", err);
    while (1);
  }
  Serial.println("Camera initialized successfully.");

  Serial.println("Initializing WebSocket...");
  Serial.print("Connecting to ws://");
  Serial.print(websocket_server);
  Serial.print(":");
  Serial.println(websocket_port);
  webSocket.begin(websocket_server, websocket_port, websocket_path);
  webSocket.onEvent(webSocketEvent);
  webSocket.setReconnectInterval(5000);
  Serial.println("Setup completed");
```

```
}

void webSocketEvent(WStype_t type, uint8_t *payload, size_t length) {
  switch (type) {
    case WStype_DISCONNECTED:
      Serial.println("WebSocket disconnected");
      break;
    case WStype_CONNECTED:
      Serial.println("WebSocket connected successfully");
      break;
    case WStype_TEXT:
      {
        if (payload == nullptr) {
          Serial.println("Received null payload");
          break;
        }
        String reply = String((char*)payload);
        if (reply == "YELLOW_DETECTED") {
          Serial.println("Yellow circle detected!");
        } else if (reply == "NO_YELLOW") {
          Serial.println("No yellow circle detected.");
        } else {
          Serial.print("Unknown reply: ");
          Serial.println(reply);
        }
      }
      break;
```

```
    case WStype_ERROR:
      Serial.println("WebSocket error");
      break;
    default:
      break;
  }
}

void loop() {
  webSocket.loop();

  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi not connected");
  } else if (!webSocket.isConnected()) {
    Serial.println("WebSocket not connected");
  } else {
    Serial.println("Capturing frame...");
    camera_fb_t *fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      delay(1000);
      return;
    }

    Serial.println("Sending frame...");
    if (webSocket.sendBIN(fb->buf, fb->len)) {
      Serial.printf("Sent %d bytes\n", fb->len);
```

```
    } else {
      Serial.println("Failed to send image");
    }

    esp_camera_fb_return(fb);
  }
  delay(1000);
}
```

Лістинг Г.1. Код ESP32-CAM

```
import asyncio
import websockets
import cv2
import numpy as np
import logging

logging.basicConfig(filename='detection.log', level=logging.INFO,
format='%(asctime)s - %(message)s')

async def detect_yellow_circle(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_yellow = np.array([20, 100, 100])
    upper_yellow = np.array([30, 255, 255])
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    kernel = np.ones((5, 5), np.uint8)
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
```

```
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        (x, y), radius = cv2.minEnclosingCircle(contour)
        area = cv2.contourArea(contour)
        perimeter = cv2.arcLength(contour, True)
        if perimeter == 0:
            continue
        circularity = 4 * np.pi * area / (perimeter * perimeter)
        if radius > 30 and 0.7 < circularity < 1.2:
            return True, (int(x), int(y)), int(radius)

    return False, None, None

async def handle_websocket(websocket, path):
    print(f"New WebSocket connection from {websocket.remote_address}")

    try:
        async for message in websocket:

            nparr = np.frombuffer(message, np.uint8)
            frame = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
            if frame is None:
                print("Failed to decode image")
                await websocket.send("ERROR")
                continue
```

```
            is_yellow, center, radius = await detect_yellow_circle(frame)
            if is_yellow:
                print(f"Yellow circle detected at {center}, radius {radius}")
                logging.info(f"Yellow circle detected at {center}, radius {radius}")
                cv2.circle(frame, center, radius, (0, 255, 255), 2)
                await websocket.send("YELLOW_DETECTED")
            else:
                print("No yellow circle detected.")
                logging.info("No yellow circle detected.")
                await websocket.send("NO_YELLOW")

            cv2.imshow('Received Image', frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

    except websockets.ConnectionClosed:
        print("WebSocket connection closed")
    finally:
        cv2.destroyAllWindows()

async def main():
    server = await websockets.serve(handle_websocket, "192.168.0.165", 8000)
    print("WebSocket server started on ws://192.168.0.165:8000/ws")
    await server.wait_closed()
```

Лістинг Г.2. Код Python серверу